



REPLY TO
ATTENTION OF

DEPARTMENT OF THE ARMY
US ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VIRGINIA 22333-5600

2

AD-A222 292

PERI-BR

MEMORANDUM FOR Marketing and Publication

SUBJECT: Certification and Transmittal of Final Manuscript

1. The enclosed final manuscript is submitted.
 - a. Title: Semi-Automatic Methods of Knowledge Enhancement
 - b. First author: Donald Michie
 - c. Contributing author(s):
 - d. Field unit/tech area: European Scientific Coordination Office
 - e. Present FU/TA/HQ office chief: Dr. Milton S. Katz
 - f. Project name:
2. Checklist has been completed.
3. It is to be published as a: Research Note
4. The DoD Distribution statement is: Approved for Public Release;
Distribution Unlimited
5. First-time-distribution list requested is:
Additional author copies requested: _____.

6 Encls

1. Package checklist
2. Peer review - 1
3. Report documentation page
(DD Form 1473)
4. Table of contents
5. Body of the manuscript
6. Reference list


MICHAEL KAPLAN
Director, Basic Research

DTIC
ELECTE
MAY 25 1990
S D CS D

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS NONE		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) R&D 4369A-RB-01			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION The Turing Institute		6b. OFFICE SYMBOL (If applicable) PERI-BR		7a. NAME OF MONITORING ORGANIZATION ARMY RESEARCH INSTITUTE OFFICE OF BASIC RESEARCH	
6c. ADDRESS (City, State, and ZIP Code) George House 36 North Hanover Street Glasgow, G1 2AD, SCOTLAND				7b. ADDRESS (City, State, and ZIP Code) 5001 EISENHOWER AVENUE ALEXANDRIA, VA 22333-5600	
8a. NAME OF FUNDING / SPONSORING ORGANIZATION ARMY RESEARCH INSTITUTE		8b. OFFICE SYMBOL (If applicable) PERI-BR		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER C- DAJA45-86-0047	
8c. ADDRESS (City, State, and ZIP Code) OFFICE OF BASIC RESEARCH 5001 EISENHOWER AVENUE ALEXANDRIA, VA 22333-5600		10. SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO. 6.11.02.B		PROJECT NO. 201611 02B74F	
		TASK NO.		WORK UNIT ACCESSION NO.	
11. TITLE (Include Security Classification) (U) Semi-Automatic Methods of Knowledge Enhancement					
12. PERSONAL AUTHOR(S) Donald Michie; Jean Hayes-Michie					
13a. TYPE OF REPORT FINAL		13b. TIME COVERED FROM 9/86 TO 12/88		14. DATE OF REPORT (Year, Month, Day) December 1988	
				15. PAGE COUNT 31	
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	automatic structuring; automatic concept formation, chess endgame; expert systems, inverse resolution; knowledge acquisition; knowledge synthesis; machine learning; structured induction; ultra-complex domain		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The objectives of the project are concerned with the representation of human concepts in machines. The issues addressed are: 1) how to automate the development of concepts from examples supplied by an expert. 2) how to automate the validation of factual databases which have been generated by a machine. 3) how to automate the development of concepts summarising the information in comprehensive databases of raw facts. 4) how to achieve (3) above for problem domains so complex that no effective human concepts exist, or can be constructed by expert brains ("Ultra-Complex Domains" or UCD's). 5) how to develop machine methods for imparting these concepts to human experts, thus providing a method for mastering what was previously impenetrable. For reasons of economy and ease of control, the chose problem domains were chess endgames. Substantial results have been achieved with automatic structuring of problems and the design of primitive "starting" concepts, previously the province of the human intellect. The investigators believe that this work has forced an entrance to territory new to computer science. (CONT ON REVERSE SIDE)					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Milton S. Katz/Michael Kaplan			22b. TELEPHONE (Include Area Code) (202) 274-8722		22c. OFFICE SYMBOL PERI-BR

PUBLICATION CHECKLIST

Use for: Research Report, Technical Report, Research Product, Research Note, Special Report, or book published by ARI.

Type of document (circle one): RR TR RP **(RN)** SP book

Submit an original and one copy of:

Documentation required for publication

☒ 1. Certification DF, signed.

Peer review - 1

☐ 2a. Recommended changes made.

☐ 2b. Changes not made--reviewer's name not to be used.

Peer review - 2 (only one required for Research Note)

☐ 3a. Recommended changes made.

☐ 3b. Changes not made--reviewer's name not to be used.

Letter(s) of permission to quote copyrighted material

☐ 4a. Required and submitted.

☒ 4b. Not required.

Sensitivity review

☐ 5a. Required and submitted.

☒ 5b. Not required.

Security review

☐ 6a. Required and submitted.

☐ 6b. Not required.

☒ 7. DD Form 1473, completed.

Manuscript

Foreword

☐ 8a. 6.3 research.

☐ 8b. Other research funding.

☒ 8c. Not required (Research Note).

Acknowledgment

☒ 9a. Included.

☐ 9b. Not included.

Executive summary

☐ 10a. Included.

☒ 10b. Not required (RN, RP, or book).

☒ 11. Table of contents (with Lists of Tables and Figures).

Accession For	
NTIS	CRA&I
DTIC	TAB
Unannounced	Justification
By _____	
Distribution / _____	
Availability Codes	
Dist	Avail and/or Special
A-1	

Publication Checklist (cont)

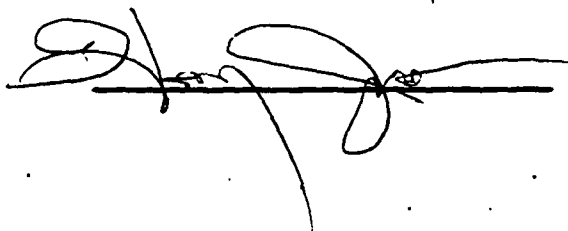
Body of the manuscript

- ☒ 12. Text of report (with Tables and Figures).
 - a. All pages are present and numbered properly.
 - b. Text is print-ready copy.
 - c. Table of contents accurately indicates configuration of document.
- ☒ 13. Reference list (documents listed are cited in text).

Appendixes

- ☒ 14a. Included (they are necessary explanatory information).
- ☐ 14b. None prepared.

Signature of individual completing checklist.

A handwritten signature in black ink, consisting of a series of loops and a long horizontal stroke, positioned above a solid horizontal line.

PEER REVIEW
(Subject matter expert review)

Date Due: _____

Date Received: 12/1/88

Date Returned: _____

Reviewer (full name) Michael Kaplan

Organization Office of Basic Research

Manuscript Title Semi-Automatic Methods of Knowledge Enhancement

Author or COR Donald Michie

I. RATINGS (extent to which criteria are met):

A. FOREWORD, EXECUTIVE SUMMARY (BRIEF): Are they clear? Are they consistent with the contents? Are they directed to target readers or users? Do they concisely highlight the important findings?

No ☐ Moderately ☐ Yes ☐ Substantially ☐

B. INTRODUCTION, BACKGROUND, OBJECTIVES: Is the literature review relevant to the research conducted (necessary and sufficient)? Is the statement of the problem pertinent to the target audience? Is it clear? Is it supported by concrete data or evidence?

No ☐ Moderately ☐ Yes ☒ Substantially ☐

C. APPROACH, METHOD: Is it appropriate? Was there a valid experimental design and data collection plan? Are they competently described and were they competently executed?

No ☐ Moderately ☒ Yes ☐ Substantially ☐

D. RESULTS: Are they clearly presented? Is there appropriate use of tables and figures? Were proper statistics used? Were they used correctly?

No ☐ Moderately ☒ Yes ☐ Substantially ☐

E. DISCUSSION AND CONCLUSIONS: Do they follow from the data and literature review? Are they comprehensible? Are conclusions and recommendations warranted and usable by intended audience?

No ☐ Moderately ☒ Yes ☐ Substantially ☐

Peer Review (cont)

II. Recommendations

☐ Should not be published.

☐ Return for reconsideration (e.g., reanalysis, additional data collection, or rewrite).
Should not be published as is. (Comments may be made in Section III or as an enclosure.)

☐ Publish after minor revisions. (Comments may be made in Section III or as an enclosure.)

☒ Publish as is.

My name ~~may~~ may not appear on the inside cover as reviewer.



Reviewer's Signature

III. Comments on any of the above ratings or recommendations.

SEMI AUTOMATIC METHODS OF KNOWLEDGE ENHANCEMENT

**By: Donald Michie
Jean Hayes Michie**

Final Report on DAJA45-86-0047

**Prepared for: The US Army Institute for the
Behavioural Sciences, through its European
Science Coordination Office in London**

December 1988

SEMI-AUTOMATIC METHODS OF KNOWLEDGE ENHANCEMENT

19. KEY WORDS: automatic structuring, automatic concept formation, chess endgame, expert systems, inverse resolution, knowledge acquisition, knowledge synthesis, machine learning, structured induction, ultra-complex domain.

20. ABSTRACT

These experiments are the most recent in a series started in January 1984. The objectives of the project are concerned with the representation of human concepts in machines. The issues addressed are:

- (1) how to automate the development of concepts from examples supplied by an expert.
- (2) how to automate the validation of factual databases which have been generated by machine.
- (3) how to automate the development of concepts summarising the information in comprehensive databases of raw facts.
- (4) how to achieve (3) above for problem domains so complex that no effective human concepts exist, or can be constructed by expert brains ("Ultra-Complex Domains" or UCD's).
- (5) how to develop machine methods for imparting these concepts to human experts, thus providing a method for mastering what was previously impenetrable.

For reasons of economy and ease of control, the chosen problem domains are chess endgames. Substantial results have been achieved with automatic structuring of problems and the design of primitive "starting" concepts, previously the province of the human intellect.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SEMI AUTOMATIC METHODS OF KNOWLEDGE ENHANCEMENT		5. TYPE OF REPORT & PERIOD COVERED FINAL REPORT SEPTEMBER 1986 to DECEMBER 1988
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) DONALD MICHIE (Principal Investigator) JEAN HAYES-MICHIE (Research Associate)		8. CONTRACT OR GRANT NUMBER(s) DAJA45-86-0047
9. PERFORMING ORGANIZATION NAME AND ADDRESS Turing Institute, George House, 36 North Hanover Street, Glasgow, G1 2AD, SCOTLAND		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS ARI SCIENCE COORDINATION OFFICE, EUROPE European Research Office, US Army, London, England (Scientific Coordination Office: Dr Milton S Katz)		12. REPORT DATE DECEMBER 1988
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Prepared under Contract Number DAJA45-86-0047 US Army Research Institute for the Behavioral and Social Sciences, 5001 Eisenhower Avenue, Alexandria, Virginia 22333, USA		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) See attached.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) See attached.		

19. ABSTRACT (CONT)

in which the mental labor of specifying and programming is largely shifted from the human to the computer. In particular problems previously too complex to program have been rendered accessible, and a methodology established for generalizing this result.

(ii)

CONTENTS

INTRODUCTION AND PREVIOUS WORK

Expert systems	2
The need for inductive systems	2
History of inductive systems	3
Previous ARI-funded work	4
Automatic structuring	5

RESULTS

Database construction and cognitive studies	5
Machine studies	6

FUTURE WORK	10
-------------	----

CONCLUSIONS	10
-------------	----

ACKNOWLEDGEMENTS	11
------------------	----

REFERENCES	11
------------	----

APPENDICES:	1. Database construction
	2. BBQ: Grandmaster concepts
	3. DUCE and CIGOL

PUBLICATIONS

INTRODUCTION AND PREVIOUS WORK

Knowledge lies all around us in the brains of experts of various kinds. Lacking: fast ways of getting it out for inspection and acquisition, not to mention incorporation into machine systems.

Potential knowledge of even greater mass and quality lies all around us in the world's databases. But no-one has any way at all of mining it's hidden meaning, except in the relatively weak and uncertain form of statistical correlations.

The design of software tools to reduce such data to structural forms which reveal its most meaningful subdivisions is a major objective of the project. Further, we aim to design tools which can themselves act as tutors to human experts in domains so complex that they are beyond unaided human grasp.

To summarise: the aims of the project are as follows:

- (1) how to automate the development of concepts from examples supplied by an expert.
- (2) how to automate the validation of factual databases which have been generated by machine.
- (3) how to automate the development of concepts summarising the information in comprehensive databases of raw facts.
- (4) how to achieve (3) above for problem domains so complex that no effective human concepts exists, or can be constructed by expert brains ("Ultra-Complex Domains" or UCD's).
- (5) how to develop machine methods for imparting these concepts to human experts, thus providing a method for mastering what was previously impenetrable.

The project thus aims to create a new sub-category of the class of software known as expert systems. The new kind combines expertise in the UCD itself with expertise in how to instil this expertise into a human partner.

Expert systems

The software technology of expert systems began in the late 1970's, with the aim of implementing computing systems which would perform convincingly as advisory consultants.

It was initially thought that computer-based consultation would be confined to the conventional performance goal of earlier computing technologies, namely to deliver good answers to the client's input questions. This turned out **not** to be the whole picture.

First: The client demands explanations as well as answers.

Second: The system is typically required to have facilities for improving and refining its knowledge from tutorial interaction with the client's own domain specialists.

Third: An additional feature has come to the fore whereby the system can be made to generate improved codifications of domain-specific knowledge for human use. This has become known as "knowledge refining".

Our objective is the construction of a fully automated facility for the synthesis, refinement and validation of rule-structured representations of useful knowledge **in forms which are interpretable by both computer systems and human brains**. Without this "principle of symmetry"[1], the movement towards the automatic handling of socially critical functions (e.g. air-traffic control, nuclear accident warning etc) is seen as dangerous.

The need for inductive systems

The tradition of knowledge engineering as it has evolved in the United States has been based on a scenario in which the knowledge engineer labours hand in hand with a domain specialist whose knowledge it is desired to transfer. The engineer seeks to draw the required expertise out of the expert's head in the form of rules which can be encoded in the machine system.

The knowledge engineer's objective is to convert human know-how into "say-how". Once in this form the traditional arts of programming and compiling can convert it into machine code i.e. the machine representation of the know-how. At run time it generates "machine show-how", in other words the expert behaviour that the customer desires.

This seemed a promising way to go because experts are confident about their ability to access the large body of pattern-based rules which they have in their heads. Except in rare and special cases, this confidence is ill founded. Furthermore it seems that the more complex the mental skill, the greater the proportion of it which is encoded in intuitive form and hence beyond access by anybody, including the expert. One reaches the dark area of inaccessibility surprisingly soon as one moves up the complexity scale. We must therefore find some way of going from human know-how to machine know-how other than by the method of articulation.

When an expert is asked to teach his skill to a human apprentice, most of the work is done by presenting a cleverly graded and sequenced series of *tutorial examples*.

It thus appears that there is a way, other than by explicit articulation, of moving this conceptualised material into another agent provided that the agent is in a suitably prepared state i.e. is capable of learning by example. To use this technique with computers, we require effective algorithms for **inductive inference** to be executed by the recipient machine. Such algorithms must simulate the apprentice's ability to reconstruct from the tutorial examples a mental model of the master's skill.

History of inductive systems

There is a rich history of inductive inference work in artificial intelligence, dating from Earl Hunt's Concept Learning System in the 1960s [2].

Later Ross Quinlan extended Hunt's algorithm and implemented his own algorithm, ID3 [3], [4]. This is the basis of all commercially viable induction system at the present time. ID3 is given a training set of positive and negative instances, where each instance is represented as a fixed list of attributes or properties. It is required to produce a decision tree for differentiating positive and negative instances of some decision class. Since a decision tree is logically equivalent to a conditional expression in a programming language, we can say that the output of the algorithm is a program. The synthesized expert program can then be run on new material, to test the level of skill induced by the particular examples used as the training set.

In our laboratory, we subsequently developed an enhancement of ID3 to include the ability to handle numerical as well as logical attributes of the problem domain. Expert systems already developed with this approach include Dow-Jones forecasting, error-message interpretation in UCSD Pascal, classification of lymphatic cancers, evaluating test firings of the shuttle main engine and many others [eg 5]. A valuable side-effect of building inductive systems became apparent. The rules produced automatically from the tutorial examples were found to aid dramatically the users' understanding of the domain.

Previous ARI-funded work

In a series of experiments funded by ARI, between 1984 and 1986 we followed up this line of work, extending it to an investigation of the use of inductive systems as teaching tools for children, using elementary algebra as the problem domain [6], [7]. Results of the pilot run were positive.

The possibility emerged that inductive methods could not only help us understand domains where no adequate understanding existed previously: they could do the same where no adequate understanding or human skill could exist, due to the domain's combinatorial complexity. Such problem areas are here called "ultra-complex" domains, and there are many examples in e.g. routing and placement problems, multi-channel signal interpretation, multi-variable process control. Techniques for rendering such areas comprehensible would have obvious applications in both industrial and military areas. Evolving such techniques is the long-term aim of this series of experiments.

We saw as a potentially ultra-complex domain the chess endgame King and two Bishops versus King and Knight (BBN). We investigated the 'complexity rating' in parallel with the algebra experiments. A fully computed-out database for this ending was made available to us by Dr Ken Thompson. The world's foremost endgame scholar, A J Roycroft, studied the BBN endgame and his performance was later tested, [8], [9]. Results showed that Roycroft was indeed unable to master the entire domain, but that he had achieved at least partial mastery. This decided us to turn our attention to an ending considered to be even more inaccessible to the human brain, King and two Bishops versus King and Queen (BBQ).

Automatic structuring

The structuring of problems by the invention of intermediate sub-concepts for forming top

level descriptions and also the design of the primitive "starting" concepts is currently the province of the human expert in the construction of all expert systems, either by dialogue acquisition or by rule induction over examples. However it is clear that if we are to tackle ultra-complexity, these two human-dependent processes must both be automated. Early progress on these problems has already been reported. This year our project has followed this up.

RESULTS

In 1988 experiments have been progressed along the following lines: (1) **Database construction and cognitive studies** which include the construction of our own interactive database to act as ORACLE, and a study of Grandmaster concepts in this ending, and (2) **Machine studies** of construction of software tools for automatic structuring of raw data.

Database construction and cognitive studies

(1) Database construction: The problem space of BBQ is some hundred million positions, allowing for symmetries. It is therefore possible to compute out the entire ending, and depending on the interactive interface provided it is possible to return a definitive answer (an oracle) to questions such as: Is this position won for White, with White to move? How many moves to solution? What is the best move? We were fortunate in having the co-operation of Ken Thompson who made his database available. However, an oracle, like all software, needs to be validated. The only known method for validating such a database is to check it against another constructed from scratch. We therefore undertook the task of building our own BBQ database. This proved to be difficult, first because of a shortage of disc space for mounting both the Thompson database and our own. The former was supplied on tapes incompatible with hardware at the Institute and successfully loading them onto our system was a task taking many months. Through the generosity of ARI, we were able to buy a 300MB Eagle magnetic disc, which made the storage of these databases a practical possibility, but transferring our existing systems from the VAX to the Sun Workstation, which offered greater flexibility of access, uncovered discrepancies between the two systems. These took five months to debug. Methods of access to the database also provided problems, now largely solved. The system adopted to maximise efficiency of access has now been tested and it has been verified that the paging system of memory allocation works correctly. Details are given in Appendix 1.

(2) Study of Grandmasters concepts in BBQ.

Thirteen Grandmasters were asked to comment on a maximum-length path to solution taken from the database, in the sense of breaking the solution path into "phases of play". Response was patchy. Apparently awed by the complexity of the problem only 3 GM's responded and all asked for no public use to be made of their comments.

The list of prospective subjects, together with the problem set them and the three anonymous solutions are given in Appendix 2.

Machine Studies

Here progress has been extremely good. Dr Muggleton in his algorithms DUCE and CIGOL, has produced the first fully successful algorithms for automatic structuring. Descriptions of the DUCE operators have been given in earlier reports and a summary is given in Appendix 3. The work has been proved correct, and during the last year experimental trials have been made in domains as different as chess and brain lesion diagnosis [10]. Recent work is described below.

Referring to the "Aims of the Project" listed at the outset of this document, progress with item 3 has dominated the project's closing six months. This item demands *total automation of the concept-formation process*, meaning by this the extraction from completely raw data of complete and correct human-type concepts without the need to supply the concept-learning algorithm with any "attributes", "sub-concepts" and the like; all these the system must invent or discover for itself.

Since nothing of the sort has been as yet accomplished, or even to our knowledge attempted, by any laboratory, we thought it best to start at the "easy" end. We chose as a test of CIGOL's powers (with an accompanying test series using the DUCE algorithm for comparison) a relatively clear and simple chess concept, namely the "legality" of a position in a King-Rook-King (KRK) endgame.

Illegal positions in KRK

We chose as the experimental domain the set of all possible combinations of the three pieces (White King, White Rook and Black King) on the squares of the board. Not all of these combinations are legal chess positions in a KRK endgame. If we consider only white-to-move positions, then a position in the domain is illegal when either

1. one or more of the three pieces is on the same square, or
2. the two kings occupy adjacent squares, or
3. the black king is in check (since all positions are for white to move).

The goal of this learning problem may be thought of, then, as the autonomous creation of a logic program which correctly classifies positions as illegal or not according to the three conditions above. We are trying to show that such a program can be **inductively generated from examples** of individual legal and illegal positions held in a database, i.e. raw data.

Raw data representation

In any domain there is usually a lowest level description for objects within that domain. This can be determined by the lower limit of the resolution scale of sensors which filter incoming data, e.g. for a perceptual task. In other cases, such as chess, the primitive data level is given by definition. For chess, therefore, we define raw data to consist of board positions described only in terms of which pieces are on which squares of the board.

According to this definition, the size of the KRK domain is 64^3 , or 262,144 positions. Approximately one third of positions in this total space are illegal. For these experiments, example positions are chosen at random from the total space, and classified by table lookup in a white-to-move KRK database. In this database every position is classed either as illegal or legal. Legal positions are also classed as either draws or wins at a certain depth.

The actual representation for positions used in the experiments is that used by CIGOL, and is virtually identical to what is usually described as Edinburgh-syntax Prolog. All positions supplied to CIGOL in these experiments are raw data as defined above, since they describe positions only in terms of the location on the board of each piece. This representation may be seen in an excerpt from an experimental session which is shown in the Figure and discussed in the next section.

Example of sub-concept induction

To illustrate some steps in the process of induction by inverse resolution [11], we present in the Figure part of a logged session with CIGOL. In this extract we see a new relation, a two-place predicate which the user names **adjacent-ranks**, being invented from raw data.

The remainder of this section provides some explanation of the Figure.

In this Figure all CIGOL is shown in bold-face type, while all user input is in italics. CIGOL is designed to be used in a similar way to an interpreted language like Prolog, Lisp or BASIC. To this end, the CIGOL interpreter resembles Prolog in appearance although the top-level prompt (!-) is different.

Firstly, CIGOL is presented with an example of a position which is illegal because the two kings are diagonally adjacent on the board. A position is represented by a three-place predicate "illegal", or its negation. The three arguments of this predicate are, in order, the squares occupied by the White King, White Rook and Black King in the position. Each square is denoted by its file and rank, as in standard chess notation, so that square "c4" corresponds to the co-ordinate "(3,4)" from White's view of the board. In CIGOL's representation a position with the White King on square "c4" contains "wk(c,4)".

When an example is presented in response to the interpreter prompt CIGOL immediately commences searching for potentially good applications of its inductive transformation operators. A good operator application is defined as one which enables program compaction. The "..." in the figure indicates where messages from CIGOL regarding the progression of the search have been omitted for clarity. However, no generalisation by inverse resolution is possible from this first example. After each search stage has been completed, CIGOL shows the user all clauses currently in the program which is under construction prefaced with the head "I know:".

Next, the user supplies a second example of a position which is illegal due to king adjacency. CIGOL again commences searching for potentially good applications of its inductive transformation operators. If one can be found, a good operator application is presented to the user as a query which includes the name of the applied operator, e.g. TRUNCATION, and the hypothesised transformation.

At this point, CIGOL finds a generalisation of the first two examples, and shows this to the user, requesting confirmation of its correctness *in all the cases which it covers*. However, the user rejects this suggestion, since it is not the case that all KRK positions where the White King is on file "c" and the Black King is on file "d" while the White Rook is on square "h1" are illegal.

- 8A -

1 ?- *cigol*

!- *illegal(wk(c,4),wr(h,1),bk(d,5)).*

...

I know:

illegal(wk(c,4),wr(h,1),bk(d,5)).

!- *illegal(wk(c,5),wr(h,1),bk(d,6)).*

...

TRUNCATION: (-12)

Is *illegal(wk(c,A),wr(h,1),bk(d,B))* always true? *n.*

...

INTRA-CONSTRUCTION

illegal(wk(c,A),wr(h,1),bk(d,B)):-p39(A,B).

p39(4,5).

p39(5,6).

What shall I call *p39*? *adjacent.*

...

I know:

adjacent(4,5).

adjacent(5,6).

illegal(wk(c,A),wr(h,1),bk(d,B)):-adjacent(A,B).

not(illegal(wk(c,A),wr(h,1),bk(d,B))).

!- *~Z*

yes

1 ?- *~Z*

[End of Prolog execution]

FIGURE: CIGOL induces sub-concept "adjacency" concept in KRK.

Following this rejection, CIGOL re-commences the search, and subsequently finds an application of the INTRA-CONSTRUCTION operator which is presented to the user. We can see from the figure that CIGOL has begun to "get the idea" with this second question, which concerns the adjacency of the files of the White and Black Kings in both of the examples provided. This, it will be remembered, is part of the reason why the positions are illegal, since in both the Kings are diagonally adjacent.

The application of the INTRA-CONSTRUCTION operator has created a new predicate, to which CIGOL has applied its own name, "p39". Here the user is asked not to confirm the correctness of a generalisation, but either to apply a meaningful name to the newly created predicate or to reject it as not useful. In this case the new predicate is very useful as it is a partial definition of the concept of "adjacent-ranks", and is named thus by the user.

At this point, following the successful application of the INTRA-CONSTRUCTION operator, the "Adjacent-ranks" predicate name is incorporated into the set of clauses in the current program, which is again displayed on the terminal by CIGOL under the heading "I know:". Note that the negative reply given to the suggested TRUNCATION above results in the addition to the current set of clauses of the *negation* of that generalisation. This is because CIGOL now knows that illegal ($wk(c,A), wr(h,1), bk(d,B)$) is *not* always true.

The session could have been terminated at this point although other options are open to the user. These include adding further examples, or verifying the partial program developed so far by CIGOL against a test set of randomly chosen KRK positions.

The CIGOL session in the Figure illustrates the way in which CIGOL operates. In particular, this example demonstrates how the invention by machine of a pre-specified target program from raw data may be carried out. Experimental work is now proceeding on testing the performance of the DUCE and CIGOL systems on the KRK legality problem.

Correctness and comprehensibility

The native mode of the CIGOL system is interactive. This exemplifies an incremental learning approach, where examples are added in stepwise fashion to build the target program. Owing to the fact that all induced concepts are presented to the oracle, correctness and comprehensibility of the resultant program are ensured. Since the oracle is correct by

definition, representing as it does a complete specification for the target program, no incorrect program clauses will ever be added. Therefore, if the initial program is correct, then the target program will be too. Comprehensibility of the resultant program is aided by the fact that any new terms hypothesised by CIGOL are given names by the oracle which are relevant to the task domain. This can be seen in the Figure where the induced two-place predicate is named "adjacent-ranks", which is meaningful when discussing chess positions. For instance, the same relation in some other domain might have been named "successor".

FUTURE WORK

Our current and future investigations into the total automation of the concept-formation process in the KRK domain depend on running CIGOL, and DUCE for comparison, in batch mode with large example sets. Prototype batch versions of both algorithms have been developed in which the oracle has been automated. In effect, this means that every suggested operator application is automatically accepted. This usually leads to *incorrect generalisations* being incorporated into the induced programs. However, the effect of this incorrectness can be measured against test sets of examples. This performance metric provides an empirical basis for comparing implemented versions of inverse resolution. One experimental condition currently under investigation includes investigating the effects of adding items of background knowledge about the experimental domain. This has already led to re-evaluations of search control and example efficiency. Another experiment involves running both algorithms to obtain performance statistics for the induced concepts on separate test sets, using examples chosen under one condition by a pseudo-random number-generator and under another by a theoretically "perfect tutor". Implementation of a means of automating the production of "perfect tutor" example sets has pointed to an improved basis for evaluation of potential compaction over several inverse resolution steps.

In parallel, teenagers with no previous knowledge of Chess will be tested on their ability to conceptualise from a series of examples the legal/illegal concept in the KRK domain.

CONCLUSIONS

We believe this work has forced an entrance to territory new to computer science in which the mental labour of specifying and programming is largely shifted from the human to the

computer. In particular problems previously too complex to program have been rendered accessible, and a methodology established for generalising this result.

ACKNOWLEDGEMENTS

This work was carried out with the help of a grant from the US Army Institute for the Behavioural and Social Sciences, through its European Science Coordination Office in London, UK under contract DAJ-45-86-0047: the views expressed are those of the author and do not necessarily reflect official positions of the US Army. Support was also provided under the British Government's Alvey Database Demonstrator No 092/167. Facilities were provided by the Turing Institute, Glasgow, UK., Interact R & D Corporation, Victoria, BC, Canada and Intelligent Terminals Ltd, Glasgow, UK: Intelligent Terminals is part of the Infolink Group.

REFERENCES

- [1] Sergot, M. Personal Communication.
- [2] Hunt, E. Marin, J. and Stone, P. (1966). *Experiments in Induction*. New York: Academic Press.
- [3] Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples. In *Expert Systems in the Micro-Electronic Age* (ed. D. Michie), pp 168-201. Edinburgh: Edinburgh University Press.
- [4] Quinlan, J. R. (1983). Learning efficient classification procedures and their applications to chess endgames. In *Machine Learning: an Artificial Intelligence Approach* (eds. R. S. Michalski, J. G. Carbonell and T. M. Mitchell), pp 463-482. Palo Alto: Tioga.
- [5] Michie, D. (1986). The superarticulacy phenomenon in the context of software manufacture. *Proc. R. Soc. London, A*, 405, 185-212.
- [6] Paterson, AM (1984). Computer Induction in a Tutorial Context. M. Phil Thesis, University of Edinburgh.

- [7] Michie, D. and Hayes-Michie, J. (1986). Semi-automatic method of knowledge enhancement. *Final Report (Contract DAI-A45-84-C-0017)*, Glasgow: Turing Institute.
- [8] Roycroft, A. J. (1988). Expert against oracle. In *Machine Intelligence 11*, (eds J. Hayes, D. Michie and J Richards), pp 347-373. Oxford: Oxford University Press.
- [9] Michie, D. and Bratko, I. (1987) Ideas on knowledge synthesis stemming from the KBBKN endgame. *ICCA Journal*, 10 (1), 3-13.
- [10] Muggleton, S. H. (1987). DUCE, an oracle based approach to constructive induction. *Proc 10th International Conference on Artificial Intelligence*. pp 287-292. Los Altos: Kaufmann.
- [11] Muggleton, S. H. (In press). Inverting the resolution principle. To appear in *Machine Intelligence 12*, Oxford: Oxford University Press.

APPENDIX 1

Database construction

An endgame database is a file which contains values for every position in the chosen endgame. The simplest form of information stored would be the answer to the question - "Is this position won for white with white to move?". To answer this question would require one bit of storage per position. However, we may require the answer to a more complex predicate, such as, "Is this position won for white with white to move, how many moves are required and what is the best move?". The increased complexity of the predicate results in a corresponding increase in the amount of storage required. It is the amount of disk storage available that is the limiting factor on how much information can be stored for each position. All of the databases currently used at the Institute answer the question - "Is this position won for white with white to move and if so in how many moves?".

The databases currently stored on the disk are shown below. These comprise databases used for earlier work in this series, and sub-games of BBQ.

<i>Endgame</i>	<i>Positions (including illegal)</i>	<i>Size (Kilobytes)</i>
KQK	29568	57.8
KQKB	1892352	3696.0
KBBKN	121073664	118272.0
KQKBB	121044992	118272.0

Installation of KBBKN and KQKBB Databases

The two large databases referred to above were supplied to the Turing Institute by Ted Thompson of AT & T Bell Laboratories, Murray Hill, New Jersey, USA. Unfortunately they were supplied on tapes which were incompatible with the hardware at the Institute. After much trial and error they were eventually loaded successfully onto our systems.

A new 330mB Eagle magnetic disk was installed on the Sun 3 Workstation, to store the various chess endgame databases used in our development work.

Transfer of chess playing software to Sun Workstations

The arrival of the new Eagle disk allowed us to transfer our existing systems from the VAX to the Sun Workstations which offered greater processing power. Principle among these systems was a chess playing program developed by Alen Shapiro which used the KBBKN database. Significant problems arose in porting this system because of the different byte ordering used within the two machine systems. This together with a number of non-standard coding practices used by the original author meant that a period of some five months had to be devoted to debugging this software.

Memory Management Systems

Over the next year we hope to develop our own KQKBB database to use in our expert system research. Even with the installation of the new magnetic disk, space for new databases is still at a premium. The time required to develop a new database is also wholly dependent on the size of the database as it is this which determines the number of disk accesses required and hence the amount of time required to build a new database. In order to minimise both the disk space and computational time requires we use symmetry. The major symmetry is *king pair symmetry*. Using standard chess notation, with the bottom square of the lefthand column as a1 and the top square of the righthand column as h8, we define an octant a1-d4-a4-a1. The white king is restricted to this area. This is a valid restriction as we can reflect or rotate any position of the white king so that it is placed on the above octant. The position of the black king is determined by that of the white king. The black king must

- (a) not be adjacent to the white king and
- (b) if the white king is on the a1-d4 diagonal then the black king must be rotated above a1-d4 diagonal.

These reflections and rotations give 462 pairs of legal king positions each with an associated reflection and/or rotation, which are stored in a lookup table in dynamic memory.

Another useful symmetry involves the two bishops. As they are of opposite colours, each is assigned to a 32 square subset of the board. While the value of *king pair symmetry* has been proven in the construction of three and four piece endgame, *bishop symmetry* has not yet been implemented. The possible trade off between increased processor time and decreased disk requirements by using this symmetry in test programs will be evaluated. However, for the purposes of this document it is assumed that *bishop symmetry* is used. Therefore, after the use of symmetry we have,

462 positions for the two kings,
32 positions for the first white bishop
32 positions for the second white bishop and
64 positions for the black queen

giving a range of $462 \times 32 \times 32 \times 64 = 30,277,632$ positions. The constructed database will have to answer the following predicates:

- (a) Is this position won for white, with white to move, and if so how many moves are required.
- (b) Is this position lost for black, with black to move, and if so how many moves are required.

Each of these predicates will require one byte of storage per position. Our total storage requirements will therefore be

$$30\,277\,632 \times 2 \text{ bytes} = 57.75\text{mB}$$

A further point to consider before construction of the database was the nature of access. Systems such as the playing module software written by Alex Shapiro for king and two bishops against king and knight (KNNKN) access their database entries on an individual basis. However, the large number of file accesses required during the database construction call for a different approach. During the construction of the smaller three and four piece endgames the entire database was held in dynamic memory, for reasons of processing economy. However, with a five piece endgame the size of files involved prevent this (A total of 64mB of memory would be required). It is therefore proposed to introduce a form of memory paging system which would hold the most efficient and largest possible sections of the database in memory. An explicit form of paging rather than from that provided by UNIX, has been chosen in order to keep the implementation of the algorithm broadly similar to that used for smaller endgame databases.

It is proposed to partition the database files into 462 sections of 64 Kilobytes. Each partition will hold the data for all possible positions involving a particular pair of kings. It would have,

32 positions for the first white bishop,
32 positions for the second white bishop and
64 positions for the black queen

giving 65536 possible piece arrangements including those which are illegal. It is proposed to run the construction algorithm at a low processing priority, so the number of partition in memory and then accessing the other entries individually.

In order to test the above theories, a new copy of the KQKB database was constructed using the above symmetries and memory managment system. Using this small four piece endgame it was possible to verify that the paging system of memory allocation worked correctly. In addition to this programs to test the *bishop symmetry* were developed. It was verified that there is a worthwhile trade off between the diminished disk usage and the increase processor use.

Dear,

The 5-man pawnless endgame of Queen against two Bishops is the second chess endgame where the computer has recently upset the verdict of the specialist literature, namely the books of endgame theory.

The first such endgame was two Bishops against Knight, now known always to be won by the superior side, even when the defence adopts the famous configuration due to Kling and Horwitz. As you may know, this result was first published in EG, the quarterly magazine of which I am the editor and publisher. This was in 1983. EG is now publishing detailed descriptions of how this endgame can be won. This learning from the computer has been achieved by prolonged interaction with the 'total knowledge' data base computed out by Ken Thompson of Bell Labs.

The same process will eventually reveal the secrets of the still mysterious Queen against two Bishops endgame. For the present, however, we have a parallel point of departure, namely a single example of best play for both sides in a winning solution of maximum length. It is to assist in this process of discovery, to accelerate the learning process, that I am inviting you, as an acknowledged expert and interested party, to participate in a combined chess-and-psychological information-gathering experiment.

You will find enclosed the original published sample of play in the two Bishops against Knight endgame, with my own attempted description of five phases in that solution. You may use this sheet as a guide, or you may ignore it.

The play you are invited to examine is on a second sheet that gives the 71 moves of the winning play in the ending Queen against two Bishops.

Please identify and describe the major features of the play in this ending as you see them. The features may be 'phases of play', or they may not. Whatever you observe, please write your observations down clearly and return them to Mrs Jean Michie in the enclosed envelope.

That is the essential part of the experiment. If you wish to do more, please feel free to do so. That 'more' might be to suggest underlying principles of play, to indicate, and even to suggest names for, patterns that occur (or are avoided), to suggest questions to which we need answers, and in short to make any constructive comments at all.

This invitation is being distributed simultaneously to a dozen talented master analysts in several countries. Significant chess results will appear in EG. Significant psychological results, such as may relate to how we invent new and useful problem-solving concepts, may be submitted for publication to a journal of high academic repute at the discretion of Mrs Jean Michie, cognitive psychologist at the Turing Institute in Glasgow, where Donald Michie is Chief Scientist. The project is being funded jointly by the Turing Institute and the US Army Research Institute for the Behavioural and Social Science in Europe.

Please send your descriptions and comments, if any, by the end of April, 1988. If you do not already see EG and would like to see the published results of this new style of experiment, please write to me.

Good luck, and thank you for cooperating!

Please accept my best wishes for 1987.

Yours sincerely

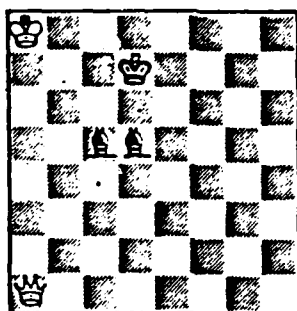
John Roycroft

QUEEN against TWO BISHOPS

maximum length win: 71 moves

position and moves taken from the ix.1986 issue of the
Journal of the International Computer Chess Association.

A selection of chess masters, analysts and endgame specialists
has been invited to describe and comment on the play in
their own words. Please refer to the covering letter.



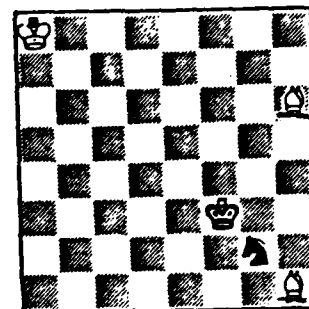
1 ♖b8 ♕d6† 2 ♖a7 ♕c5† 3 ♖a6 ♕c4† 4 ♖a5 ♕d6 5 ♜c1 ♕d5 6 ♜c3 ♕d4 7 ♜f3†
♕e5 8 ♜g3† ♕e4 9 ♜h4† ♕e3 10 ♜e7† ♕d3 11 ♖b4 ♕d5 12 ♜e1 ♕f3 13 ♜h4
♕e3 14 ♖c4 ♕e2† 15 ♕d5 ♕f3† 16 ♕e6 ♕e4 17 ♜g3† ♕d2 18 ♜f4† ♕d3 19 ♕d6
♕c3 20 ♜g3† ♕d2 21 ♜g1 ♕d3 22 ♕c5 ♕c2 23 ♜e3 ♕b2 24 ♖b4 ♕c3† 25 ♖a3
♕b2† 26 ♖a4 ♕c3 27 ♜c5 ♕e4 28 ♜c4 ♕f3 29 ♖b5 ♕d1 30 ♜a2† ♕d3 31 ♜g2
♕e2 32 ♕c5 ♕d1 33 ♜h3† ♕d2 34 ♕d5 ♕b3† 35 ♕e4 ♕c2† 36 ♕f4 ♕d4 37 ♜a3
♕d3 38 ♜b4† ♕c3 39 ♜a4 ♕b2 40 ♕f3 ♕c3 41 ♜a7 ♕b2 42 ♜c5 ♕e2† 43 ♕f2
♕d3 44 ♜g5† ♕c2 45 ♜d5 ♕d2 46 ♜a2 ♕c3 47 ♕e1 ♕c4 48 ♜a5† ♖b3 49 ♜b6†
♕c3 50 ♜b7 ♕c2 51 ♜h7† ♖b3 52 ♜b1 ♕c3 53 ♜e4 ♕a3 54 ♜e3† ♖b2 55 ♕d1
♕b3† 56 ♕d2 ♕b4† 57 ♕d3 ♕a2 58 ♜e2† ♖b3 59 ♜c2† ♖a3 60 ♕d4 ♕b3 61
♜c1† ♖a2 62 ♕d3 ♕a3 63 ♜c7 ♕b2 64 ♜a7† ♖b1 65 ♕d2 ♕d5 66 ♜b6 ♕e4 67
♜b5 ♕g2 68 ♜e2 ♕b7 69 ♜f1† ♖a2 70 ♜f7† ♖a3 71 ♜x b7

Fig. 2

The position from BBQ, together with the 71 moves to win, sent to 13 chess masters
for analysis.

The five phases of play in the endgame two bishops against knight, based on the description by AJR in EG, xi.1983.

1. In a maximum length solution position, either wK or wB is under restraint. This restraint must be lifted, which may take a dozen moves. In the case of wB under restraint the releasing manoeuvre involves a long wK march: with wBh1 it is necessary to play wK to f1 or to h3.
2. In the next phase B1 has freedom to retreat in good order, but is slowly forced out of the centre of the board. He chooses to adopt the defensive formation due to Kling and Horwitz (1851). This is based on bN adjacent to a corner square (but not in the edge) with bK in attendance. ca. 14 moves.
3. W manoeuvres to create one of only four positions known where bK is ejected from the 'fortress' without the possibility of reforming in any corner. ca. 8 moves.
4. Phase 4 is complex, fluid, lengthy and difficult. B1 strives for maximum freedom and frequently seems to be on the verge of success. It takes W over 20 moves, not to be found in any textbook, and characterised at times by excruciating slowness and mystery, before bK, having failed time and again to repeat the Kling and Horwitz position, ends up on the board's edge near a corner and accompanied by bN.
5. The remaining 12 moves or so show bN being lost, whether staying close to bK or running away.



1. d1f8 2. d6d+ 3. f3 4. a7 5. f2 6. d2 7. f1 8. b7 9. d2 10. c6 11. f1 12. d6 13. c4 14. d3 15. e3 16. d2 17. d6 18. e3 19. d5 20. f5 21. e5 22. d5 23. b6 24. d1 25. d2 26. e3 27. d2 28. f4 29. d2 30. b3 31. d2 32. e3 33. d3 34. d5 35. d5 36. e4 37. f5 38. d4 39. d3 40. e4 41. d5 42. d1 43. f6 44. d5 45. d3 46. e5 47. d1 48. d5 49. d4 50. d6 51. b3 52. f6 53. d3 54. d4 55. d1 56. d3 57. d5 58. e5 59. e6 60. d4 61. f6 62. d6 63. h7 64. d5 65. e7 66. d8 67. d4 68. e6 69. f6#

Fig. 3- Description by the endgame scholar, A. J. Roycroft, of the phases of play of BBN. to act as an exemplar for the 13 chess masters invited to attempt an analysis of the more complex BBQ ending.

LIST OF MASTERS

1. Yochanan AFEK
2. IGM Pal BENKO
3. David V HOOPER
4. Paul A LAMFORD
5. FIDE Master Graham D H LEE
6. IGM Dr John D M NUNN
7. IM Dr Enrico PAOLI
8. Axel ORNSTEIN
9. Erkki PUHAKKA
10. IGM Jon SPEELMAN
11. IGM Jan TIMMAN
12. Alain VILLENEUVE
13. N KARAKLAIC

Replies Received

The following 3 reports are presented without attribution, following authors' wishes:
Reports on maximum-length win (71 moves) in BBQ.

S 1.

"...It is, as all the endings investigated by the computers, very hard to understand.

There seems to be no special phases of the play, but there are some special features.

1. White manouvres to avoid the black drawing position Kb2, Lc2, Lc3, but the black pieces are allowed or forced close to it.
2. The win sometimes depends on zugzwang.
3. The queen often suddenly moves far away from the black pieces to control important squares - 13. Dh4, 21. Dgl, 41. Da7, 50. Db7, 63. Dc7.
4. The white king is not very active in the winning process. The queen more or less by herself forces the black pieces to the side of the board.

Finally on the question - how does white win after 33 - Kc2 or 34. Kc2".

"Besides the moves submitted to me I have additional information from the computer, of which I shall make use. Firstly, I have a note regarding alternative W and B1 moves at each turn to play with an indication of the results that they lead to. Secondly I am informed that the position WHh5, WQe8, B1Kg7, B1Bf6, B1Bf5 is a zugzwang, and that there are no other zugzwangs. A zugzwang here indicates that White to play would draw, Black to play would lose. i.e. whoever is to move is at a decisive disadvantage. The existence of this zugzwang appears to confirm analysis carried out by del Rio and Lolli in the 1760s. This is a version of the position they give called the Lolli position: WKg4 WQe6 B1Kg7 B1Bf6, B1Bg6. Black to play, 1...Bh7 2 Qd7+ Kg8 3 Qe8+ Kg7 4 Kh5 Bf5 (zugzwang). Thus the Lolli position is drawn. If this final position were moved one file to the left (East) then B1 to play loses (as he would in the Lolli position) in this case by 1...Bg7 2 Qc7+ Kf8 3 Kg6. The computer indicates that the position is not a zugzwang, so W to move cannot obtain a worse result than would occur with B1 to move; therefore W wins. I infer that Black's only drawing position is the Lolli position, and, perhaps, a few 'proto-Lolli' positions. I exclude draws by repetition of moves, pins, skewers, mere tactical devices that White can normally avoid and which, therefore, have little bearing on the problem before us. Two examples: A.) 68.Qh5? Bd4+ 69 Kg5 Be3+ &c, for if W avoids the checks by, say, Kg6, then...Be8+ leading to the capture of White's Q. B.) 70 Qa3? Bh2+ 71 Kc5 Bg1+, and if the WK moves away from the checks B1 gets the Lolli position.

The method of winning or attempting to win this and several other similar kinds of endgame were discovered empirically long ago. The B1K must be driven to the edge of the board so that threats of checkmate may be added to White's other threats. The means towards this end are also well known: getting the WK to centre so that B1K is driven back, triangulation to lose the move (a K manoeuvre). Attacking the bunch of B1 pieces from two fronts (e.g. WK from east, WQ from west), varying the direction of such attacks, the avoidance of a series of checks which drive back WK, the use of forks, skewers, and, most importantly, pins, and last, but far from least, the use of the squeeze. In the case under review, White must not permit the Lolli position. Phase 1, ending 5...Kd5, is trivial, W merely dodges checks.

Phase 2 ends 24 Kd5. This shows the moving of WK to the centre, and could be achieved by any good player. After 5...Kd5 the WK is confined to two squares on the a-file. We say that the bishops form a barrier, holding off WK, and this is a defensive resource common to all such endings. White, of course, cannot win unless the Q assists the WK, so the WK must be advanced. White achieves this by setting up a squeeze by, in this case 6 Qf4, and we see that B1, forced to move, must lower the barrier, i.e. 6...Bd3 7 Qd2 - the pin which

forms a large part of the W tactics - 7...Ke4 8 Qc3 - holding both bishops under attack 8...Bd4 9 Qe1+ Kd5, and WK advances, 10 Kb4. We see that a squeeze forces B1 to give ground - indeed, in so far as every move B1 makes leads him nearer to defeat, we might say that every W move sets up a squeeze. We cannot define squeeze precisely, and use the term, generally whenever there seems to be an 'obvious' squeeze. After 10...Bf5 White plays 11 Qe7! changing the direction of the Q's attack so that now it attacks from north and WK attacks from West. These changes of direction of attack form a prominent feature of White's play. (e.g. for the more dramatic changes see 18 Qh4+ and 19 Qd8+, 25 Qe2, 39 Qb7 and 41 Qf3+, 43 Qg8). The purpose of these changes of direction is to disorganize the defence: B1 can set up barriers of a kind, but not in orderly and maintainable fashion. A typical squeeze is 13 Qd2, mimicked one file further east with 16 Qe2. The first phase is not difficult because the WQ has a great deal of manoeuvring space of the east side of the board, but things become more difficult when Wk reaches the centre.

Phase 2 lasts from move 24 to move 60, and is, as one might expect, the most difficult phase, which might well defeat a good player. Black, now on the east side, may seek a Lolli position in either the NE or SE corner. To prevent both possibilities W attack with WK from south, WQ from North. Thus WK moves away from the centre; this will, eventually, allow B1K to return to the centre, and White must make sure that when this happens the bishops are not so placed that a good barrier defence can be organized. On move 44, any move of the WQ (other than Qf8+) would allow B1 to obtain a Lolli position, so the B1K can make its way back to the centre, but B1 is not well enough organised to prevent B1K being driven to the N edge. Note the use of triangulation to thrust the move upon B1: 27 Kc5 and Kd4 (instead of Kd4 in one move), 44 Ke3 and 45 Kf3, 50 Kg5 and 52 Kf4 and 53 Kg4.

I have shown several kinds of manoeuvre, and have noted (but not here) a few others. None that I have noted were not known before. It is possible that the computer has shown some previous unknown manoeuvres, and that for just this reason I have failed to perceive them. Perhaps I could have made a deeper study, which, I feel, would benefit from some more examples of play.

The final phase begins when BlackK, forced to the edge, is unable to get off again (60...Ke8). This disintegrative phase shows a bishop moving a distance of root 18 squares (measured from square-centres) from B1K (63...Bc5). Now we know the end is near: throughout the previous play B1 has kept the bishops close to BK. If the B moves away then it may be lost because of a fork by Q, but in any case the need to avoid a fork is likely to disorganize the B1 game.

I have described the play in terms of linked and short manoeuvres, which is the way players

S 3

"I would summarise the winning plan as follows -

- a) extricate the White King while preventing the B1KBB running together to a corner (10 moves)
- b) slowly squeeze the KBB from each side (33 moves)
- c) push them towards the corner, like a dog sending sheep into a pen, without allowing them to set up the fortress position (28 moves)

Indeed the whole solution is like one man and his dog with the queen doing enormous work (the dog) and the king keeping observation (the man)".

think. It is possible to conceive of the entire 70 moves as one extended manoeuvre, and I am inclined to this view. The best human players, however can hardly foresee more than a tenth of this distance, and will never equal the tactical depth of a computer. Thus, in 'translating' for humans we 'invent' phases and short manoeuvres, but are they really there? From the players' point of view, of course, it is not necessary to find the shortest solution, but rather to know some organized procedure to which they can work.

What has the computer taught us? In this case its most important contribution is to tell us that this endgame can be won except for the Lolli position, which was previously uncertain or unknown. A player, armed with this knowledge will make efforts to win, with some success perhaps, where he/she might previously have been inclined to concede a draw. I have never seen this endgame in an actual game, but now that we have this information the endgame will probably occur, on occasion. The tendency for nature to copy art, as it were, has been observed before when theorists have discovered new endgame facts.

The computer will show us a winning continuation, but not a drawing continuation, unless very many consecutive questions are asked. I have, I believe, discovered a ('proto-Lolli' position and this could be checked, but the computer will not find for me other 'proto-Lollies' - if they exist. Nevertheless, I believe the computer can yet tell us more. The computer uses many manoeuvres sequentially, but I cannot discern any pattern as to how a player might find the manoeuvres in right sequence - quite possibly there is no such organized pattern (this I suspect). I do think the example before us would increase a player's awareness of the possibilities.

To teach, we shall devise patterns, real or imaginary. We may devise these in sufficient detail and number that a player is able to win or draw, as the case may be. Players will never win positions in as few moves as the computer uses, but perhaps this is no great matter from a sporting point of view".

APPENDIX 3

DUCE and CIGOL

Using 6 operators, on an input of examples in the form of low-level conjunctive rules, DUCE is an algorithm which produces hierarchical concept descriptions from large numbers of such examples, building new higher-level attributes from existing lower-level ones. Each is automatically tested against the user (acting as oracle) for comprehensibility: i.e. is the given higher-level attribute really a "concept"? DUCE uses a set of transformations of propositional Horn clauses (the logic form in which "conjunctive" rules are conveniently expressed) which successfully compress the example material on the basis of generalisations and the additions of new terms.

CIGOL, DUCE's successor algorithm, goes further and is able to range over the entire universe of predicate logic. CIGOL's fundamental reasoning step is equivalent to inverse resolution, resolution being the fundamental step in deductive reasoning on which all modern logic programs are based. CIGOL's key attainment, impossible within the constrained propositional framework of DUCE, is in inventing new *sub-concepts* as necessary on its path to autonomous creation of the main concept. This is the same as the "design of primitive starting concepts" mentioned above as previously dependent wholly on the creating of the human problem solver.

DUCE Operators

The six operators which are used progressively to transform subsets of the rulebase are described below. The rules are expressed in propositional logic. Questions referred to the Oracle are shown in italics.

1) Inter-construction. This transformation takes a set of rules such as

$$X \Leftarrow B \& C \& D \& E \quad (1.1)$$

$$Y \Leftarrow A \& B \& D \quad (1.2)$$

and replaces them with the rules

$$X \Leftarrow C \& E \& Z? \quad (1.1')$$

$$Y \Leftarrow A \& Z? \quad (1.2')$$

$$Z? \Leftarrow B \& D \quad (1.3')$$

for example, in a blocks_world planner

$$\text{task 1} \Leftarrow B_on_A \& get_B \& C_on_B$$

$$\text{task 2} \Leftarrow C_on_B \& B_on_A \& get_A$$

$$Z? \Leftarrow B_on_A \& C_on_B$$

what shall I call Z? stack.

the new rules are

$$\text{task 1} \Leftarrow \text{stack} \& get_B$$

$$\text{task 2} \Leftarrow \text{stack} \& get_A$$

$$\text{stack} \Leftarrow C_on_B \& B_on_A$$

2) Intra-construction. This is simply the distributive law of boolean equations, which takes a group of rules all having the same head, such as

$$X \Leftarrow B \& C \& D \& E \quad (2.1)$$

$$X \Leftarrow A \& B \& D \quad (2.2)$$

and replaces them with

$X \Leftarrow B \ \& \ D \ \& \ Z?$ (2.1/2.2)

$Z? \Leftarrow C \ \& \ E$ (2.3)

$Z? \Leftarrow A$ (2.4)

for example

$\text{city} \Leftarrow \text{pop_1m} \ \& \ \text{palace} \ \& \ \text{parks} \ \& \ \text{cathedral}$

$\text{city} \Leftarrow \text{castle} \ \& \ \text{pop_1m} \ \& \ \text{parks}$

$Z? \Leftarrow \text{palace} \ \& \ \text{cathedral}$

$Z? \Leftarrow \text{castle}$

what shall I call $Z?$ sights

- 3) Absorption. Given a set of rules, the body of which is completely contained within the body of the others, such as

$X \Leftarrow A \ \& \ B \ \& \ C \ \& \ D \ \& \ E$ (3.1)

$Y \Leftarrow A \ \& \ B \ \& \ C$ (3.2)

we hypothesise

$X \Leftarrow Y \ \& \ D \ \& \ E$ (3.1')

$Y \Leftarrow A \ \& \ B \ \& \ C$ (3.2)

for example

$\text{ship} \Leftarrow \text{sails} \ \& \ \text{mast} \ \& \ \text{flag} \ \& \ \text{hull} \ \& \ \text{rudder}$

$\text{ship} \Leftarrow \text{hold} \ \& \ \text{bows} \ \& \ \text{stern} \ \& \ \text{propellor}$

the new rules are

$\text{ship} \Leftarrow \text{sailing_ship} \ \& \ \text{hull} \ \& \ \text{rudder}$

$\text{ship} \Leftarrow \text{hold} \ \& \ \text{bows} \ \& \ \text{stern} \ \& \ \text{propellor}$

$\text{sailing_ship} \Leftarrow \text{sails} \ \& \ \text{mast} \ \& \ \text{flag}$

- 4) Identification. We attempt to replace a set of rules which all have the same head, the body of at least one of which contains exactly one symbol not found within the other rules, such as:

$$X \Leftarrow A \& B \& C \& D \& E \quad (4.1)$$

$$X \Leftarrow A \& B \& Y \quad (4.2)$$

by the rules

$$X \Leftarrow A \& B \& Y \quad (4.2)$$

$$Y \Leftarrow C \& D \& E \quad (4.3)$$

for example

$$\text{gas} \Leftarrow \text{an_lt_50} \& \text{bp_lt_200} \& \text{colourless} \& \text{odourless} \& \text{non_reactive}$$

$$\text{gas} \Leftarrow \text{an_lt_50} \& \text{bp_lt_200} \& \text{inert}$$

the new rules are

$$\text{gas} \Leftarrow \text{an_lt_50} \& \text{bp_lt_200} \& \text{inert}$$

$$\text{inert} \Leftarrow \text{colourless} \& \text{odourless} \& \text{non_reactive}$$

- 5) Dichotomisation. We try to replace a set of rules with positive and negative heads, and which all have some common symbols within the bodies, such as:

$$X \Leftarrow A \& B \& C \& D \quad (5.1)$$

$$X \Leftarrow A \& C \& J \& K \quad (5.2)$$

$$X \Leftarrow A \& B \& C \& L \quad (5.3)$$

with the rules

$$X \Leftarrow A \& C \& Z? \quad (5.1')$$

$$X \Leftarrow A \& C \& Z? \quad (5.3')$$

$$Z? \Leftarrow B \& D \quad (5.4')$$

$$Z? \Leftarrow J \& K \quad (5.5')$$

$$Z? \Leftarrow B \& L \quad (5.6')$$

for example

$$\text{country} \Leftarrow \text{monarch} \& \text{parliament} \& \text{capital} \& \text{borders}$$

$$\text{country} \Leftarrow \text{president} \& \text{borders} \& \text{constitution} \& \text{capital}$$

$$\text{country} \Leftarrow \text{governor} \& \text{capital} \& \text{police} \& \text{borders}$$

the new rules are

country «= capital & borders & government

country «= capital & borders & government

government «= monarch & parliament

government «= president & constitution

government «= governor & police

6) Truncation. This operator generalises by dropping conditions. A set of rules which all contain the same head, such as:

$X \ll A \& B \& C \& D$ (6.1)

$X \ll A \& C \& J \& K$ (6.2)

is replaced by

$X \ll A \& C$ (6.1/6.2)

for example

fish «= gills & fins & scales & large

fish «= gills & scales & small & tails

the new rule is

fish «= gills & scales

The decision of which operator to apply is approached through "best-next" search. Since each operator can reduce the number of symbols in the rule-base, we search for the application which produces the largest symbol reduction, i.e. we apply Occam's razor.

In CIGOL the operators are based on precisely the above six, but with various elaborations appropriate to CIGOL's comparatively enriched domain of predicate, as opposed to propositional, logic.

PUBLICATIONS

The following titles contain work made possible by the grant under Contract DAJ-45-86-0047 from the U S Army Research Institute for the Behavioral and Social Sciences, through its European Science Coordination Office in London.

Papers 1, 2 and 3 were prepared during the period of the previous grant, under contract No DAJA45-84-C-0017. The published versions of 1 and 2 are now attached, together with a revised version of 3, now in press. Publication No. 4 is attached; 5 is not yet complete.

1. Roycroft, A J (1988). Expert against oracle. In *Machine Intelligence 11*, (eds J Hayes, D Michie and J Richards), pp 347-373. Oxford: Oxford University Press.
2. Muggleton, S H (1988). Inductive acquisition of chess strategies. In *Machine Intelligence 11*, (eds J Hayes, D Michie and J Richards) pp 375-389. Oxford: Oxford University Press.
3. Muggleton, S H (In press). Inverting the resolution principle. To appear in *Machine Intelligence 12*, Oxford: Oxford University Press.
4. Muggleton, S H (1988). A strategy for constructing new predicates in first order logic. In *Proceedings of the Third European Working Session on Learning* (ed. D Sleeman) pp123-130, London: Pitman.
5. Michie, D and Bain, M (1989). Machine acquisition of concepts from sample data. To appear in *Artificial Intelligence and Intelligent Tutoring Systems: 1989 Spring Symposium at the University Maine*.

Expert Against Oracle

A. J. Roycroft*

The Turing Institute,
Glasgow, UK

Abstract

A computer-generated combinatorial data base that plays optimally an almost undocumented and very difficult five-man chess endgame (i.e. the data base can be considered as an oracle) was matched against a domain specialist who had prepared for the contest with minimal prior access to the data base. His preparation and strategy are described and the results of the contest itself briefly summarized. The paper closes with an illustrated discussion of the selected endgame in Master practice.

1. PURPOSE OF PROJECT

The chess endgame specialist, in contrast to the tournament player, assesses his skill against the criteria of perfection, whether in adjudicating a position or in selecting a move. Automatic construction of complete look-up tables (data bases) by computer makes it possible to apply these exacting criteria in practice. By use of such an 'oracle' Kopec and Niblett (1980) were able empirically to verify the present author's claim to have acquired high-level mastery (in the actual test the play was move-perfect) of the play of won-for-White positions of king and rook against king and knight, an endgame of which full knowledge was lacking prior to the creation of a data base. The purpose of the new project is to investigate the problems and nature of skill-acquisition in an endgame selected as being so complex as to lie beyond the power of the unaided human endgame specialist to master thoroughly. For this purpose the author selected the ending king-bishop-bishop-king-knight (BBN). BBN was historically assumed to be a draw from a general position until at my suggestion Mr Ken Thompson computed an exhaustive BBN data base. The existence of the data base, and samples of its output, were published in the international quarterly magazine *EG* (Thompson and Roycroft, 1983).

* Present address: 17 New Way Road, London NW9 6PL, UK.

The Thompson data base contains complete information on how the two-bishops side can win from all but a few elementary or bizarre positions in a total space of some 250,000,000 positions. It thus exhibits 'skill' at a demonstrably unsurpassable level in a domain that is of formidable difficulty and complexity for human experts.

This characteristic makes possible in principle the absolute measurement of human performance in a difficult domain, and offers the opportunity to explore how 'inert but absolute' knowledge can be accessed and adapted for teaching, for learning, and for the development and validation of expert systems that aim to perform as well as the data base—but without it.

We also for the first time have the possibility to explore thoroughly a significant endgame data base at 'super-expert' level. Future combinatorial data bases in chess and in other domains may be more massive still. Early experience of handling them will be of value.

The present paper reports measurement of the performance of a domain specialist before (Part 1) and after (Part 2) he had been allowed unlimited access to the oracle data base, from which, for any legal position the user can retrieve the following:

- (i) all optimal single-ply (that is, at a depth of one white or black move) continuations; and
- (ii) the length of the remaining optimal path.

The author is a lifetime student of the chess endgame. He is the author of *The Chess Endgame Study* (1972 and 1981) and edits and publishes the international magazine *EG*. He is a strong, but not master, chessplayer. He is acknowledged worldwide as an endgame specialist.

2. THE TASK OF THE DOMAIN SPECIALIST (IN PART 1)

In October 1984 the author undertook:

1. To study the five-man pawnless chess endgame of (white) king and two bishops (one on light squares, one on dark squares) against (black) king and knight, using any available aid except the data base itself, with the exception of a set of 12 variations already excavated and provided in 1983 by Mr Thompson from the data base illustrating optimal play from one of the 32 worst-case-for-White (two bishops) starting positions. Time limit imposed on this period: none.
2. To record as fully and faithfully as possible all thought processes (the dated record to include time taken, chess positions and moves, sources of information used, discoveries, errors, corrections, trains of thought, going over previously trodden ground, etc.).
3. To announce when ready to face the data base, i.e. when no greater mastery of the material seemed achievable by private study.
4. To take the white (two bishops) side against the data base without

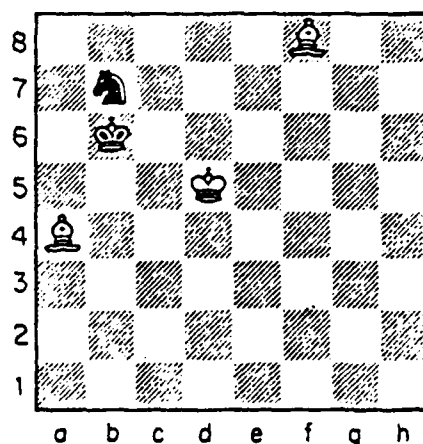
prior preparation of the particular positions to played and to play under strict tournament conditions (moves to be timed, and no moves taken back), with two exceptions: analysis on a separate board to be allowed (the domain specialist is not a practising chess master used to tournament conditions but an endgame scholar accustomed to analysing with board and men, similar to the situation that obtains in correspondence chess); and the so-called '50-move rule' to be disregarded.

The confrontation of domain specialist and oracle data base concludes Part 1. If, as is expected, the human performance is sub-optimal, Part 2 follows, in which the specialist is now allowed access to the data base. A second confrontation or test, with different positions but the same number of them, concludes Part 2.

3. BACKGROUND

The earliest known reference in chess literature to the pawnless endgame king and two bishops against king and knight is in the middle of the nineteenth century (Kling and Horwitz, 1851, pp. 62-5). R1, the first of two positions given by the authors, is the more important as it is largely independent of the positioning of the white force. It is given by them without supporting analysis but with the statement that the bishops 'cannot win if the weaker side can obtain a position similar to the above, but they win in most cases'. The second position, a win, is then given with a solution and a number of supporting variations extending to 14 moves. One or other of both positions is repeated in the subsequent literature up to 1983 (e.g. Pachman, 1983, pp. 19-20), with no modification to the verdict.

The author (Roycroft, 1972, p. 207) raised a doubt about the correctness of the claim that R1 (and positions like it) cannot be won. This doubt was confirmed in 1983 by output from the Ken Thompson data base.



R1. Kling and Horwitz (1851). Either side to move.

The data base was generated by a method already known in principle (Ströhlein, 1970). First, all possible positions of checkmate (with the given force), and all positions where the knight is safely captured (without subsequent stalemate), are automatically generated. These comprise the finally won positions that the side with the bishops aims for, and at the same time they are the positions that the side with the knight wishes to postpone as long as possible. From this starting 'position set' the first 'derived set' of positions can be generated, the set of White to Move (WTM) positions that are 'Won in 1'. By an essentially similar, but logically more complex, process the antecedent Black to Move (BTM) position sets are generated and marked where and when appropriate 'Lost in 1'. The basis of an iterative 'maximin' or 'backing-up' procedure has now been established, whereby the solved depth increases in principle by one ply (one white move or one black) per pass. This iteration is initiated and relentlessly pursued until no more positions can be classified. At this stage all won positions will be marked with the solution depth. For a more detailed description of the process see, for example, Roycroft and Niblett (1979) and Thompson (1986). Residual positions still unmarked will be drawn, illegal, or, in a microscopic number of BTM instances, won for the knight's side. In Ken Thompson's solution only WTM positions are stored, the BTM positions being generated when required by program: for convenience we refer simply to the 'data base' whether WTM positions only or both WTM and BTM positions are physically stored.

The results have been widely reprinted in the world's other chess magazines. However, guidance in the domain literature as to how this endgame should be played remains (August, 1985) restricted to paraphrases of the sentence of Kling and Horwitz quoted above, that is that the defending side should always aim for a position like R1, because it is the only safe draw. (At the end of this paper we give an example of the influence of this advice on practical master play.) As a result of the present research it is likely that future advice to the superior side will be to steer towards the Kling and Horwitz position, since the winning method from that position is (or rather will be) well charted. (For a list of the principal authorities on the chess endgame see the entries within parentheses in the section References. However, Averbakh, the major modern authority, does not include the two bishops against knight endgame because endings with two pieces on one side are in principle excluded from its scope.)

4. THE FIVE PHASES OF THE PAWNLESS ENDGAME TWO BISHOPS AGAINST KNIGHT

The division of a maximum length solution to this endgame into five phases has been described by the author (Thompson and Roycroft,

1983). The following is an updated version. The quoted passages are taken from the article in *EG*. Where a number of moves is mentioned this refers to consecutive optimal moves by White, the side with the bishops.

4.1. Phase number

1. In a maximum depth solution position the white force will initially be under some constraint from the black force: either the white king or a bishop will be immobilized. It may take from six to 12 moves to lift this blockade, depending on its nature.

2. In the next phase Black retreats slowly and in good order and 'seeks refuge in the Kling and Horwitz position. This may be in any corner'. This phase takes us up to move 20, approximately.

3. White's task in phase 3 is to manoeuvre in order to set up any of a small number (probably only four) of 'exit' positions, that is, exits from a Kling and Horwitz position. Black is then forced out into the open. Typically this phase lasts six or seven moves.

4. 'The next stage is complex, fluid, lengthy and difficult. Black strives for maximum freedom, and frequently seems on the verge of achieving it. It takes White some 23 moves, not to be found in any book and characterized at times by excruciating slowness and mystery, before' Black, 'having failed time and again to repeat the Kling and Horwitz position, ends up' with his king 'on the board's edge, near a corner and accompanied by the black knight.'

5. 'The remaining dozen or so moves show the knight being lost, whether he stays close to the black king or runs away.'

The longest solutions have 66 white moves. There are 32 distinct positions that have this depth, though they group into 'families' of positions.

5. THE STORY OF PART 1

5.1. The 'private study' phase

The private study phase began in October 1984. The material available for study comprised:

(i) published books (in English, German, and Russian) on the chess endgame in general. In contrast to the thriving literature on individual chess openings there is very little published on specific endgames. The books do not cover the endgame two bishops against knight in any useful depth. (See References);

(ii) ten full-length (66 moves) solutions and associated list of one-ply-deep equi-optimal moves provided in August 1983 by Ken Thompson to the author in the latter's capacity as editor of *EG* magazine;

(iii) two further full-length solutions, also from Ken Thompson in 1983, with no alternative moves;

(iv) three 66-move and 67-move full-length solutions from an independent researcher. The 67-move solutions were later shown to be faulty (Comay and Roycroft, 1984), and the correctness of the 66-move maximum optimal depth thereby corroborated;

(v) the 32 distinct positions at the maximum optimal solution length, also provided by Ken Thompson to the author in 1983, but without chess moves;

(vi) the frequency table of the numbers of WTM positions at every solution length from 66 to 1, also provided by Ken Thompson to the author.

On 18th January, 1985 the domain specialist intimated in writing his readiness to confront the data base in the test to end Part 1.

5.2. The protocol

Separate publication of the protocol record of the domain specialist's thought processes is intended. It runs to over 200 pages and will be supplemented with appendices.

5.3. The Part 1 test and summarized results

The test began on Friday, 29 March 1985. Two test sessions were aborted due to program failure, and there was a two weeks' interruption for holiday. The 10th and final test position was played on Tuesday, 30 April 1985.

Two sessions were abandoned by the domain specialist, after 70 and 69 moves respectively. The remaining eight positions of the test were won by the domain specialist, giving a 'tournament' performance of 80%. The only other measurement of his performance that is available at present to the author is the ratio of the total of the optimal solution depths of the original positions to the number of moves actually taken by the author: 38%.

Both measurements are crude. If the sessions abandoned by the specialist after 70 and 69 moves had been abandoned at the outset without any winning attempt at all, the 38% figure would increase, thereby putting a premium on early abandonment. This is, however, not the case with the refinement (Doran-Michie 'path efficiency') used by Michie (1986) in his review of these same experiments, which rates abandonment at any stage as equivalent to taking infinitely many moves. It is not clear what measurement would be least unsatisfactory. Two other measurements will almost certainly give different figures and should at least be calculated:

1. The ratio of optimal moves to sub-optimal moves in all the moves played by the domain specialist.

2. A measurement that takes into account the domain specialist's division of the endgame into five phases. This would log a minimal penalty against a move that is sub-optimal but which kept the solution within the same phase; it would log a heavier penalty against a move that set the solution back a phase; an even heavier penalty would be imposed on an error that set back the solution two phases, and so on; the heaviest penalty would be for a blunder that gave away the win. If a penalty were measured in numbers of question marks (i.e. '?'), with the lowest penalty rated at a single question mark, then a session could be aborted by prior agreement if the total of accumulated penalties (i.e. the total of question marks deserved) in a session passed a certain threshold. This content-related measurement of performance was proposed by the author but not adopted, one argument against being that the division of solution into phases is at present subjective.

The reason not all measurements are available to the author is that work is proceeding and it is considered that even such ancillary information relating to an earlier test could be of indirect assistance to the domain specialist, who, at the time of writing has not had the test to conclude Part 2. Since a major object of the combined Parts 1 and 2 is to determine how and to what extent a human specialist can be aided in his comprehension of a complex domain, such additional information might, however slightly, distort the performance and measurement. Full statistics will be reported in the planned monograph on the total experiment covering Parts 1 and 2 (Further experiments with the present oracle, and experiments with other, even more massive data bases are envisaged.)

6. THE DOMAIN SPECIALIST'S STRATEGIES

Implemented strategies are necessarily domain specific: they have to be described in chess terms. But some general remarks for non-chess players may be helpful.

6.1. For non-chessplayers

Here is no place to debate what, if anything, chessplayers have and non-chessplayers lack. But being an amateur problem-solver as well as a chessplayer the author recognizes that all problem-solvers have some common skills and motivations, whatever their specialist knowledge or favoured domain. The awe in which non-chessplayers commonly hold chessplayers of even less than master strength is based partly, if not mainly, on myth. In the interests of better understanding the present section of this paper aims to demolish two specific myths.

The first myth: enormous numbers

A frequent argument to boost the myth of the arcane genius chessplayer invokes 'enormous numbers'. The number of possible chess positions

exceeds the highest astronomical numbers; the number of possible chess games exceeds the number of possible chess positions, also astronomically. These are incontrovertible facts.

But a chessplayer does not have to remember or recognize all these positions and games, any more than any of us need remember or recognize all possible breakfasts in order to eat breakfast, or need remember or recognize all possible books in order to read a book.

Humans tame large numbers by ignoring them. Instead they seek and manipulate patterns, even if the patterns are initially tentative, approximate or unsound. It is a common-sense conjecture of everyday experience that a good pattern will have inferior patterns in its ancestry. If we persevere and are willing to learn, later patterns should be superior to earlier.

In the universal child's game of noughts and crosses (in American it is 'tic-tac-toe', in Russian 'Krestiki-noliki') how many different positions are there? Interestingly, the question is posed, and answers provided, in the literature of artificial intelligence (examples: Nilsson, 1971, pp. 137-8; Shirai and Tsuji, 1982, p. 10; Rich, 1983, p. 7; Alty and Coombs, 1984, p. 80). The usual answer given relies on the implied logic that there are nine initially empty cells to be filled, so we start with nine possibilities for the first play, leaving eight for the reply, seven for the third play, and so on. Factorial 9, or $9!$ is the (for a game like noughts and crosses, large) number: 362,880—which reduces through laws of legality and symmetry to 'three hundred or so distinct positions with which Nought (by convention the opening player) can be confronted' (Michie, 1961).

A contrasting answer results from arguing that there are three possible states for each cell: a nought, a cross, or emptiness. This gives us a ceiling of three to the ninth power (Rich, 1983), or: 19,683—and this is *before* eliminating symmetries.

Neither of these calculations impresses the *player* of noughts and crosses.

When tackling the identical question he will rather reason like this:

1. There are three rules or conventions that govern the game, and we can look on them as constraints:

- nought starts (constraint no. 1);
- play alternates between the placing of nought and cross (constraint no. 2);
- a completed row (in any direction, including diagonally) ends the game (constraint no. 3).

2. There is then a constraint of a different kind:

- elimination of all symmetries (constraint no. 4).

3. Finally, there is a constraint of a different kind again, one of experience or demonstration, namely:

- a well played game is inevitably drawn (constraint no. 5)

Constraints 1, 2, and 3 are part of the definition of the game. Constraint no. 5 amounts to the constraint of playing the game well. Constraint no. 4 is not essential but it is convenient to all parties.

The player then puts four questions based on the five constraints.

Q1: Can noughts occupy ALL FOUR corner cells?

The answer is 'no', because by constraint no. 5 there is no vacant cell for the fifth nought, needed by constraint no. 2.

Q2: Can noughts occupy just THREE corner cells?

The answer is 'yes', but in only one way.

Q3: Can noughts occupy just TWO corner cells?

The answer here has two parts: if the corner cells are diagonally opposite one another, then the remaining pair of (corner) cells must hold crosses, and any play thereafter into the centre cell will infringe constraint no. 5 again; on the other hand with a pair of cells in adjacent corners it is easy to show (by applying one or more of the five constraints) that there are just two possible distinct configurations.

Q4: Can there be a nought in one corner only or in no corner?

With a nought in just one corner cell, or in none, there is no way to satisfy all five constraints.

The player's answer to the question 'How many positions?' is therefore

1 + 2, i.e. 'three', (See R2.)

We have seen the following answers:

362,880

19,683

3

O X O
X X O
O O X

Noughts in three corners.

O X O O X O
O X O O O X
X O X X O X

Noughts in two corners.

R2. Noughts-and-crosses/tic-tac-toe/krestik i nulik. The only 'all constraints satisfied' configurations.

Which of them corresponds most closely to the reader's experience of the game?

Chess is more complex (i.e. it holds inordinately more patterns) than noughts and crosses, but humans cannot play chess well without forming, holding and manipulating patterns any more than they can play good noughts and crosses patternlessly. We shall return to this point after demolition of the second myth.

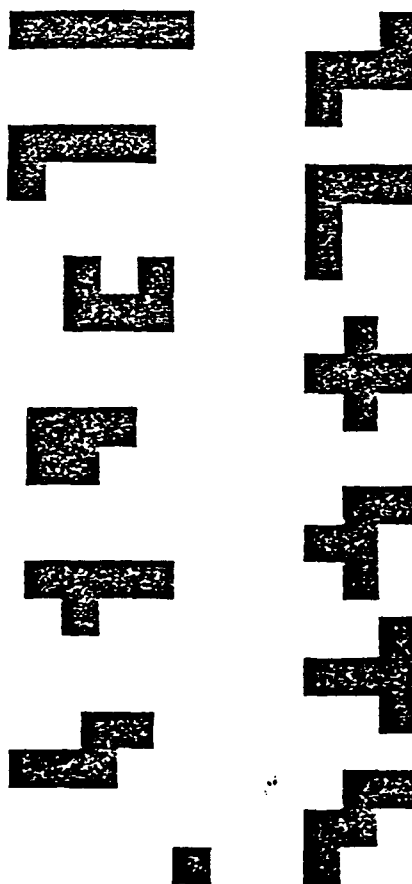
The second myth: how with all those mobile and differently moving pieces can chessplayers possibly plan?

This argument, often in paraphrased form invoking large search trees and high branching factors, requires a different counter-demonstration.

Consider the puzzle of the 'solid pentominoes'. R3 shows the 12 'flat' pentomino shapes, namely all variations on five edge-contiguous identical squares. The solid pentominoes are made out of small cubes instead of squares, but the shapes are otherwise as shown. The puzzle we shall consider is the packing of all 12 shapes into a $3 \times 4 \times 5$ unit dimensioned box. (If preferred, the target shape can be thought of as a $3 \times 4 \times 5$ 'brick' to be assembled.) As there are 60 cubelets and 60 spaces to be filled there must be no empty space and no protruding cubelet.

Let us now describe from scratch how this quite tough puzzle might be tackled.

To begin with there is no strategy. We 'play' with the pieces, trying to solve randomly, but as we play we observe ourselves, looking for a pattern. Any discerned pattern is likely to be a pattern of failure, not a pattern of success, but this does not mean that it is not a pattern, nor



R3. the 12 different plane pentominoes that can be formed from juxtaposing five unit squares.

that it will not serve. The pattern which we observe, perhaps, is that after a failed attempt one or more of the shapes in the right-hand column of our figure tend to be unused, and that these are unused more frequently than the shapes in the left-hand column. (In passing, we can ask what the left-over 'awkward' pentominoes have in common with one another? Well, a 3×3 space (like noughts and crosses—a pattern!) is necessary and sufficient to hold any one of them.) Now from a pattern to a strategy is, for a human, no long journey, albeit not always a conscious or speedy one. In the case of our observed pattern, the derived plan or strategy might be:

First fit 'several' of the 'awkward' pentominoes together into an *ad hoc* sub-assembly that will 'mutually absorb and minimize' the 'awkwardnesses' of individual pentominoes, and then arrange the remaining 'less awkward' pentominoes around the sub-assembly.

There are two important points about this strategy: it is not precise; and it is readily grasped by a human and implemented by a human, but a strategy it unquestionably is. It lends itself to objective evaluation in an experiment to compare the performance (time taken to solve, success ratios) of two groups of students, one group given the presumed benefit of the 'strategy' and the other group told nothing, but both groups tackling the identical constructional task. A complete set of the 3940 solutions to this puzzle has been computed and published (Bouwkamp, 1967).

That is typical of how the puzzle-solving mind works, whether in chess or pentominoes. Patterns are observed which lead to a strategy. A strategy will not find a solution by itself, but it serves the purpose of enabling the solver to be aware of what he is doing, to have a general aim which he can work with and refine. Such a general aim seems all the more manageable for being imprecise: 'several', 'awkward', 'mutually absorb and minimize', 'less awkward' are fuzzy terms and may be implemented differently (i.e. they may relate to different subsets of the pentominoes or to different subfeatures) by different people, though validly so: there is a distinction to be made between 'relevant fuzziness', which may even be essential in the initial communication of ideas or strategies, and ambiguity, which is to be avoided.

It is the same in chess. If you already have a strategy you can carry it out in your own way, adapting it as you go or discarding it for a better. If you don't, you can't.

If the foregoing account is accepted as valid it follows that for research purposes articulacy in the domain specialist is more important than expertise. A valuable corollary will be that a test of articulacy (to select good human subjects) can be general and standard, though designing such a test is not *a priori* the province of AI researchers.

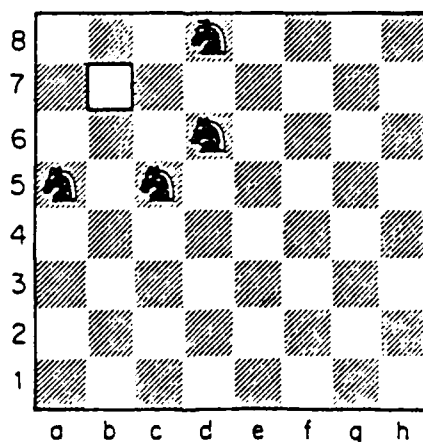
6.2. For chessplayers

The problem is strange, even to an experienced chessplayer. Classic concepts, such as the importance of the centre of the board, mobility, sacrificial combinations and positional considerations, turn out to have either no, or limited, application. A long period was spent attempting to apply long-built-in concepts of the classic type, especially those that ought to be applicable to other endgames, but eventually they were largely replaced, or modified, as a result of hard experience.

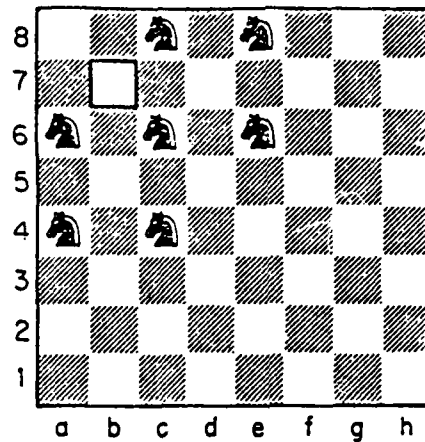
The detailed story will be told in the 200-page protocol. Here only the principal new chess concepts that were found useful will be described. However, one extant concept that proved fundamental and fruitful was the Kling and Horwitz position, though even this tried and tested 133-year-old idea was inadequate in its basic formulation: elaboration was necessary.

6.3. New chess concepts

1. *'Knight's distance' from Kling and Horwitz position.* If we consider the square b7, then ONE knight's move's distance means the four specific squares a5, c5, d6, d8. (See R4.) two knight's moves' distance means the squares a4, a6, c4, c6, c8, e6, e8, plus the 'less frequent' squares b3, d3, e4, f5, f7 (R5 and R6). The latter five squares are less frequent in the sense that bN is less likely to occupy squares that are in the centre of the board (or in sub-regions controlled by the bishops) because it is the centre of the board that White must and can control in Phase 2 of the contest, the part that consists in driving Black out of the centre. The concept of knight's distance from a Kling and Horwitz position is particularly valuable in the most difficult part of the solution, namely Phase 4. It enables us with confidence (not with certainty) to estimate how far the solution has progressed and to consider particular special strategies and tactics appropriate to that stage, rejecting (i.e. not considering) irrelevant strategies and tactics. Now the four base squares



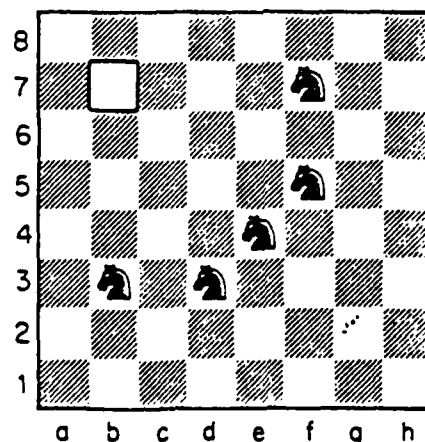
R4. Knight's distance ONE from b7.



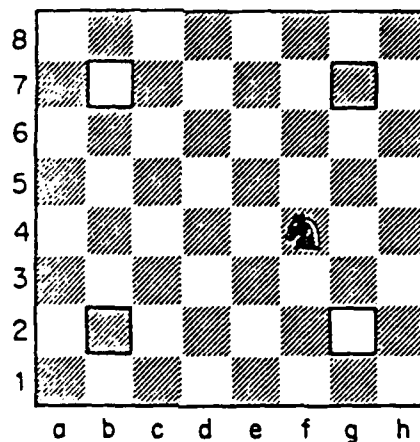
R5. Knight's distance two from b7 ('frequent' squares).

for Kling and Horwitz position mean different sets of squares in each case. But common squares begin to appear with greater frequency the greater the 'distance number'. Thus the square f4 is ONE from g2, TWO from b2 and g7, and THREE from b7—it is a 'good' square for Black's knight. (See R7.) In practice, since the black king moves slowly, and since the white force will in Phase 4 dominate the centre and some other sub-regions of the board, certain paths will be taboo to the knight: only those Kling and Horwitz positions which are accessible to the black king need be considered in applying the concept of knight's distance from a Kling and Horwitz position.

2. *Black 'king's distance' from a Kling and Horwitz position.* This concept is more static, that is, it is less liable to change from move to move, than the previous concept. The Kling and Horwitz position by definition requires Black's king; Black's king can move only to adjacent squares; it leaps to the eye when Black's king is occupying a Kling and Horwitz square; and distance simply means counting the king moves needed to reach such a square. On f4, for example, the black king is ONE from g3 (for a g2 position) and two from g6 (for a g7 position): he is



R6. Knight's distance two from b7 ('less frequent' squares).

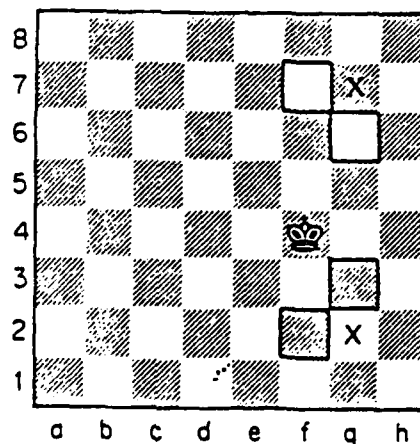


R7. Distance ONE from g2, distance TWO from g7 or b2, distance THREE from b7.

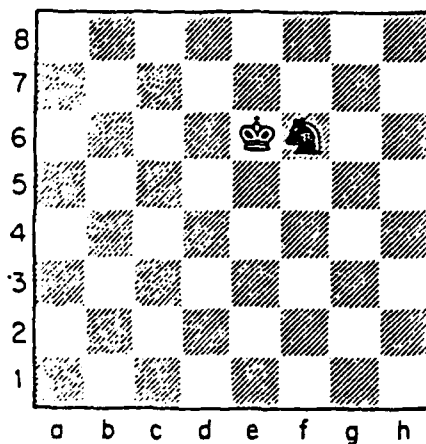
THREE from c2, but this can nearly always be ignored (in Phase 4). (See R8.)

3. *Sum of 'distances'*. This is simply the sum of the black knight's distance from a Kling and Horwitz position and the black king's distance from the same position. (The squares are, of course, different squares: a 'g7' Kling and Horwitz position implies the squares g7 for the black knight and g6 or f7 for the black king.) The summed distances are a rough-and-ready guide to progress in Phase 4. (See R9.)

4. A *'pseudo-fortress'*. In chess endgame parlance the concept of a 'fortress' is familiar. The implication is that the materially inferior side sets up a position which, due to the geometry of the chessboard is impregnable to the particular attacking potential of the superior side. The edge of the board and especially the corners are suited to fortress positions. A 'pseudo-fortress' arises when Black is evicted from a Kling and Horwitz position (which has been called a fortress, but, we now know, in error) and adopts a posture in which king and knight are alongside each other between two Kling and Horwitz positions (such as



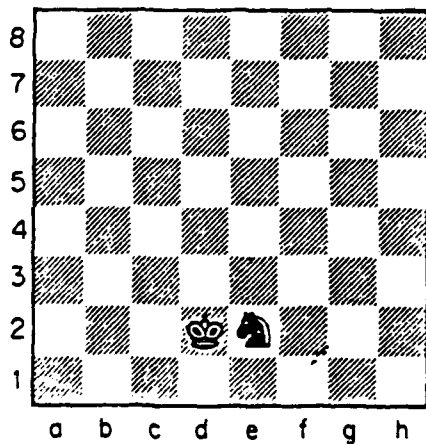
R8. King's distance ONE from g3 (a 'g2' square), King's distance TWO from g6 (a 'g7' square).



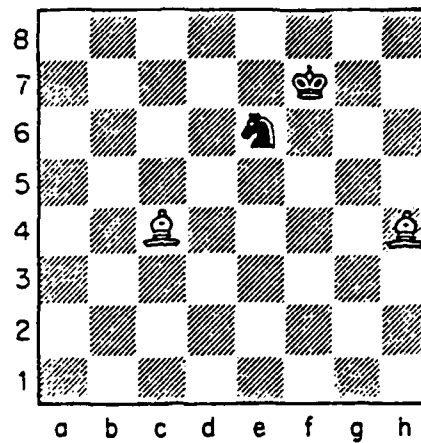
R9. Knight's distance TWO from g7, King's distance ONE from f7. Summed distance: THREE.

between b7 and g7) and at 'summed distances' of THREE or FOUR from each of them. Thus with bKd2 and bNe2 the 'summed distance' is FOUR from b2 and FOUR from g2. (See R10.) We may note, simply for the contrast, that the summed distance is EIGHT to b7 and EIGHT to g7. Such a position is strong for Black because he can adopt the strategy of oscillating to and fro with knight and king without heading for b2 or g2, which White presumably can prevent. Moreover, in his choice of an oscillating move Black, if he cannot check or usefully gain a tempo by attacking a bishop, will tend to choose a move that does not increase the summed distances. There are many manifestations of the pseudo-fortress.

5. *The 'box'.* This is the White 'counter-concept' useful in overcoming the 'pseudo-fortress'. It is a simple idea but its power is best explained by a comparison of chess diagrams and simultaneous consideration of the pseudo-fortress concept. A 'box' is a 2×2 array of squares controlled by the pair of bishops. A frequent tactic (in side-variations) to win the knight is a 'pin-crucifix' or a 'checking crucifix', which are simply special cases of the box. (See R11 and R12). Now if Black has a pseudo-fortress,



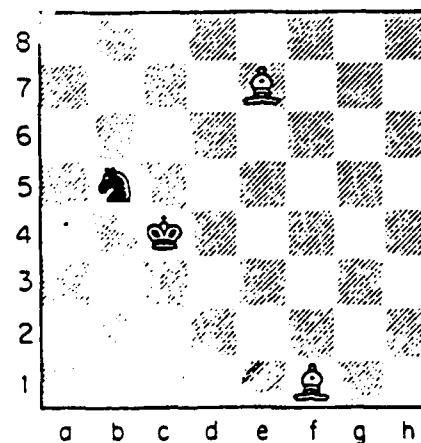
R10. A typical 'pseudo-fortress' with summed distances FOUR from 'b2' or 'g2' positions of the Kling and Horwitz type.



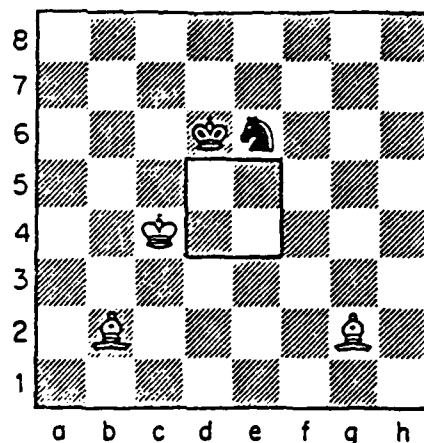
R11. A 'pin-crucifix' using a 'box' e6-f6,e7-f7. The black knight is lost without compensation.

and is happily moving king and/or knight to and fro, making sure to keep the white king at bay by checking when appropriate and then returning to the pseudo-fortress home square, how is he to be evicted? Apart from the king White has only his pair of bishops. They can be used in two obvious ways: on adjacent parallel diagonals from a distance, a classic technique, or by cross-fire to create a box. If we choose a 'box-building' strategy it is not difficult to decide what box is necessary and where the bishops must stand in order to set it up. With the black king on d2 and black knight on e2, the required box will be specified as d2-e2,d3-e3, if we assume that the white king prevents escape to c3. The result of such a box will be to drive the black king towards the cramping edge or a corner without the chance to set up a Kling and Horwitz position. (A box is not a universal panacea. It is a concept to be used with care and cunning.)

6. *'Advancing' the box.* Place the black king on d6, with the knight alongside on e6—a pseudo-fortress with summed distance THREE to b7 and THREE to g7. Place the white king on c4, and the white bishops on b2



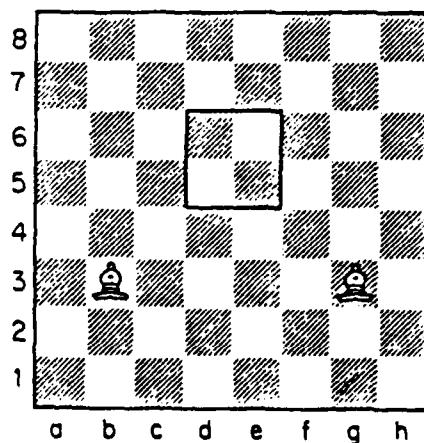
R12. A 'checking crucifix' using a 'box' b4-c4,b5-c5. The black knight is lost without compensation.



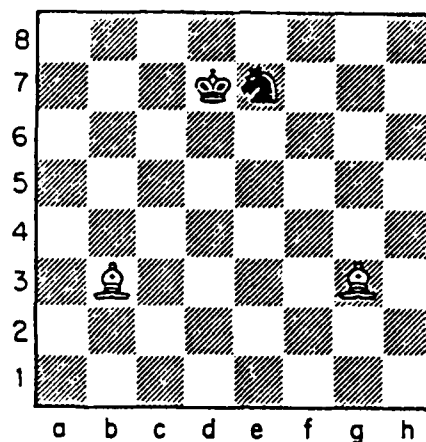
R13. Black has a pseudo-fortress. The box d4-e4,d5-e5 is ineffective. The whole box must be shifted one rank up the board.

and g2. (See R13.) The existing box (d4-e4,d5-e5) is a no-man's-land (Black cannot advance further towards the board's centre) but White's task is to drive Black out of his pseudo-fortress. If the chosen method is the box method, then the d4-e4,d5-e5 box must somehow be transformed into a d5-e5,d6-e6 box. To achieve this new box with the bishops that are able to travel only on either white squares or black squares it soon becomes evident that both bishops must switch sides of the board: the dark bishop on b2 must reach g3 (a square on the h2-b8 diagonal) and the light bishop on g2 must reach b3 (a square on the a2-g8 diagonal). (See R14.) This explains otherwise mysterious bishop moves away from the scene of action: they are unavoidable stepping-stones to where the bishops are needed to (threaten to) set up a new box.

7. *'Squinting' bishops.* Place the light bishop on b3 and the dark bishop on g3. Imagine (or place) the black king and knight on d7 and e7 respectively. (See R15.) The white king is centralized, but on no particular square, so it is omitted from the diagram. The bishops are well



R14. A box d5-e5,d6-e6, showing the bishops on opposite sides of the board compared with the d4-e4,d5-e5 box.



R15. The power of 'squinting' bishops (see text).

placed in at least five important respects: firstly, by controlling c7 and f7 they deter the black king from approaching both a b7 Kling and Horwitz and a g7 Kling and Horwitz; secondly, they are immune from tempo-gaining attack by the black king; thirdly, although not immune from attack by the knight they are relatively so, especially as the centralized white king will in Phase 4 control several squares that the knight would need to pass through to execute a bishop-harassing manoeuvre that is frequently a serious threat in Phase 2; fourthly, they create a d5-e5,d6-e6 box; and fifthly they are poised for a quick switch of sides of the board (with the probable, though not necessarily unique, purpose of advancing the box) in at most two moves (each) while still remaining relatively immune from attack (for instance the dark bishop can play to e1 and thence to b4). On the squares b3 and g3 the bishops are not 'glaring straight down' the long diagonals a1-h8 and h1-a8 but are just off-centre in this sense: hence 'squinting'. The term 'cross-eyed' graphically describes the pair of squinting bishops. The set-up is powerful and economical and it occurs frequently, with only occasionally a better square existing: for instance, the dark bishop on g3 might be vulnerable to attack or to a checking fork by the knight playing to f5, with g7 or b7 as possible defensive havens in consequence, and in this case the very remote square h2 might be superior to g3 for the bishop, but only in the short term.

7. THE STORY OF PART 2

This is the only section of the current paper to be written after the conclusion of the Part 2 test. Part 2 has no protocol corresponding to Part 1. Instead, dated files were created recording interactive sessions with the data base. Most of the time communing with the data base was spent trying out ideas, alternatives, 'what happens if' hypotheses, derived

either from the specialist's own manual record of the Part 1 test, or from attempts at a methodical approach to a problem of a particular phase of play. Considerable time was spent in examination of the output from the interactive sessions. At best such examination would answer a question; at worst it would raise further questions. Close study was made of positions around a solution-depth of 20 moves, with the hope that an increased ability to recognize such long conclusions would ease the understanding of Phase 4, where the really deep play takes place.

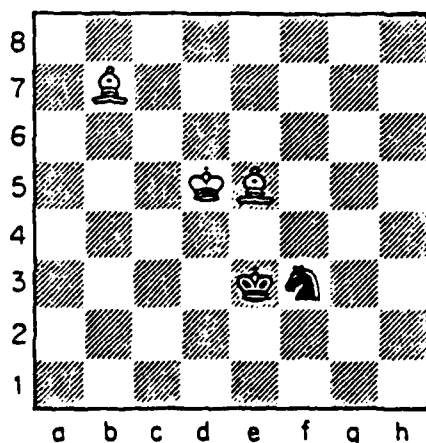
The following are some of the major new patterns to emerge.

1. The black knight is on f3 (or its symmetrical equivalent) with the black king alongside. The knight has two paths to choose from in heading for a g2 Kling and Horwitz. It is extremely unusual for both paths to be blocked. (See R16.)

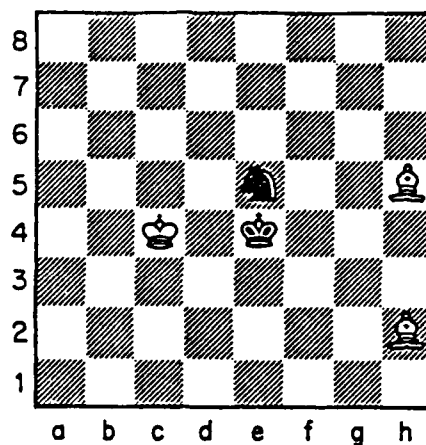
2. The pattern of white king making an outflanking move that avoids checks and covers a presumably important square to free a bishop for more important work. (See R17.)

3. Here there is a box that is prevented by the current placing of the black knight, but it is possible to attack the knight by one of the bishops. The box from Black's standpoint includes a potential outlet for his king, a 'valve', but the same square is where the knight may be lost by a pin-crucifix. This pattern might be dubbed the 'box-valve'. (See R18, R19.)

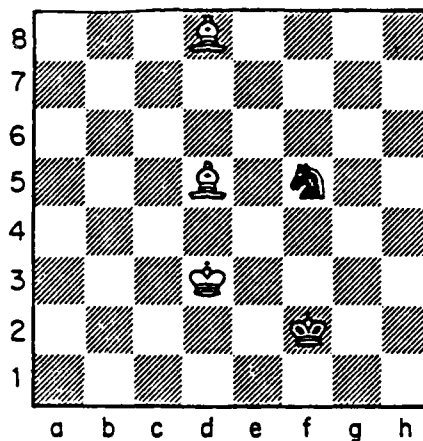
4. In the course of improving the position of the white king it 'follows' his opposite number across the board, keeping to the same or, at worst, adjacent orthogonal, especially during phase 4. In choosing the moment for such a move White must pay special regard to his king's vulnerability to checks from the knight. (See R20.)



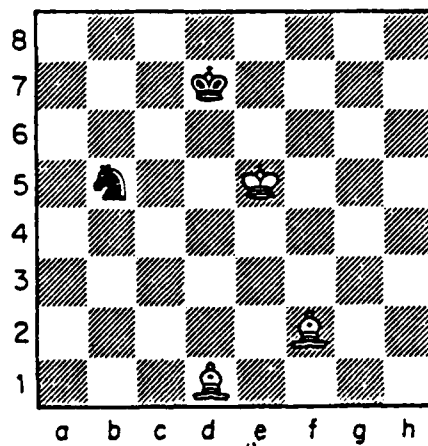
R16. White to move—depth 49. Black has two retreat paths from the square f3 to the Kling and Horwitz square g2: via h4 and via e1. If White prevents this by Bg3, then this bishop has lost its mobility and Black can threaten to set up a b2 Kling and Horwitz position. When Black can obtain a position like this it is a strong indication that the solution depth is near 50.



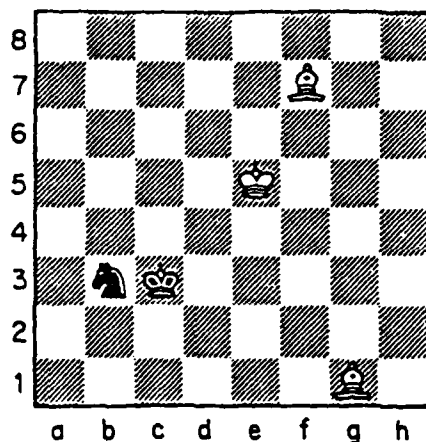
R17. White to move—depth 56. With the sequence Kc5,Nd3+;Kd6, White is safe from immediate checks and prepares to drive the black king towards the edge (any edge) starting with a bishop check on g6.



R18. White to move—depth 13. There is a latent box f3-g3,f2-g2. However, the black knight prevents the box move Bh4+. White plays Be4 and after the knight moves Bh4+ can be played.



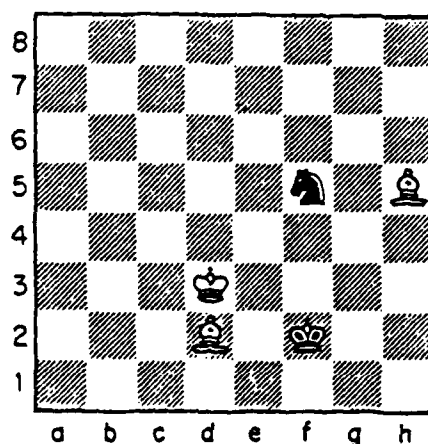
R19. Black to move—depth 19. Black's optimal move is Kc7: which allows White's reply Ke6. The explanation for not choosing Nc3; is that after Bb3,Kc6;Bd4, a box-valve position is created, with the cramping Ba4+ to come.



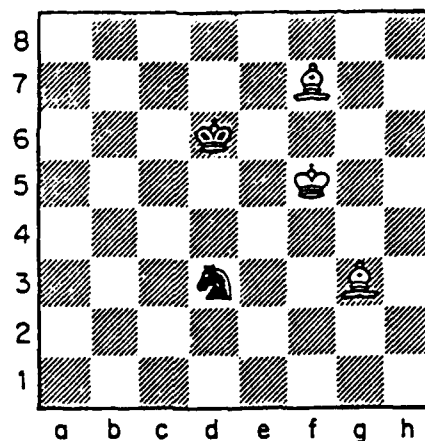
R20. White to move—depth 26. With the move Kd5, the white king is 'following' his opposite number across the board, but occupying central squares in contrast to the black king's more peripheral situation. The move also relieves a bishop of control of an important square, c4 in this case, illustrating the economy of square control by a less mobile piece when the more mobile piece is required for more active work.

5. Black has the occasional strong defence of forcing White to repeat moves as the only alternative to allowing a Kling and Horwitz. Thus a position that has the major feature of phase 4 (Black can be prevented from setting up a Kling and Horwitz) is nevertheless not a win unless progress can be made, and progress, it turns out, is only via phase 3 (a Kling and Horwitz). (See R21.)

6. The black king is forced to block a potential check by his own knight, thus allowing the white king to attack the knight with advantage. (See R22.)



R21. White to move—depth 43. In terms of the Kling and Horwitz prevention heuristic White should control the square h4 by playing Bg5. But Black then renews the threat with Kg3. Then the only way to prevent Nh4; is to play Bd2, as Nh4;Be1+ is good for White. However, in reply to Bd2, Black plays Kf2;., repeating the diagram. White will not make progress by repetition.



R22. Black to move—depth 49. Black is in check and plays his king to c5. This is optimal, despite the fact that it blocks this square for his own knight. It gives White the opportunity to advance his king with a useful gain of tempo by playing Ke4. However, and this illustrates the profundities of this endgame, the move Ke4 is not quite optimal. The sole optimal move is the mysterious Bh2.

7.1. Depthcharts

A novel technique for expressing in a concentrated visual form the content of the data base was invented. A diagram was produced showing only four of the five chessmen explicitly. The fifth man was added in the form of a number on each of the squares which the 'missing' man could legally occupy, having regard to which side was presumed to have the move. Such diagrams have been provisionally dubbed 'depthcharts'. A depthchart may identify localities where the depth is (for a white depthchart) significantly low, in which case one may confidently conjecture that if the missing white man is not in that area he ought to head towards it. The technique seems promising, but exploiting it methodically was not possible during the preparation period.

7.2. The part 2 test

The second test comprised, like the first, 10 positions.

Number of positions won: seven

Number of positions abandoned: two

One position is unaccounted for: in this, after 105 moves, when the depth of solution was low (nine) a data base move was erroneously executed on the board following the verbal notification over the internal telephone. The consequence was the data base (black) move 'king takes bishop' when the expert's board showed the knight on the square occupied, according to the system, by the black king. This was the only confusion of this kind in all the play of both tests. Performance measured by number of moves played divided by total of optimal lengths: 51.4%.

8. THE ENDGAME IN MASTER PRACTICE

This endgame has occurred several times in the tournament and match practice of the last 25 years but to the best of the author's knowledge the only serious and prolonged attempt to win a deep-solution position took place in the game between International Master Jozsef Pinter (Hungary) and International Grandmaster David Bronstein (USSR) at the international tournament at Budapest in 1978. (See R23; Benko, 1984.)

A comparison of the moves with the optimal moves (obtained by consulting the data base at every step of the game score) is of interest to chessplayers and to non-chessplayers.

Many observations and conjectures are possible arising from the comparison. However, firm conclusions are another matter. We draw none.

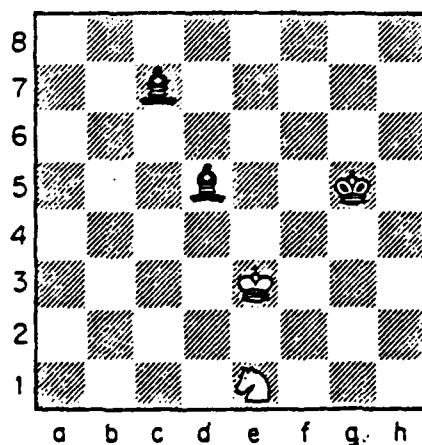
All moves played by the two masters from the moment this endgame appeared on the board are given below. The data base assumes that White has the bishops, in accordance with the normal convention of chess endgame theory. Out of respect for the valiant players we keep to the original game colours.

1. The numbering 68–117 corresponds to the serial move numbers of the actual game.

2. Where there is no move in parentheses the played move is optimal.

3. A move in parentheses is optimal, with the implication that the immediately preceding move played in the game is sub-optimal. Note that where there is more than one optimal move, only one is given. In many cases there is only one optimal move, and the occurrence of 12 equi-optimal moves for Black's 95th is unusual.

4. The two-digit number in parentheses after each black move is the optimal depth *after* that move has been played.



R23. Pinter (Hungary) vs. Bronstein (USSR) Budapest, 1978. White to play—position after Black's 67th move.

EXPERT AGAINST ORACLE

It follows that optimal play by both sides is identified by consecutive moves without any move in parentheses, when the depth will decrease by one at every move by Black.

The initial WTM position is lost for White in 54 optimal moves.

White	(optimal)	Black	depth	(optimal)
68.Kd4	(Nf3+)	Bf7	(52)	
69.Nd3	(Ke4)	Kf5	(50)	
70.Kc3	(Nc1)	Ke4	(43)	
71.Nb2		Be5+	(42)	
72.Kc2		Bg6	(42)	(Kd4)
73.Kb3		Kd5	(44)	(Kd4)
74.Na4	(Nc4)	Kc6	(49)	(Ke4)
75.Nb2	(Kc4)	Kb5	(38)	
76.Nd1		Bf7+	(37)	
77.Kc2		Kb4	(36)	
78.Kd3		Bg6+	(35)	
79.Ke3		Kc5	(45)	(Kc4)
80.Nf2	(Kf3)	Kd5	(41)	
81.Nh3		Bd4+	(40)	
82.Kf4		Be8	(48)	(Bg7)
83.Ng5		Bb6	(50)	(Bg7)
84.Nf3		Bc7+	(50)	(Bc6)
85.Ke3		Bg3	(50)	(Bc6)
86.Ng5		Bh5	(49)	
87.Nf3		Bc7	(50)	(Bd6)
88.Kf2	(Ne1)	Ke4	(41)	
89.Nh4		Kf4	(40)	
90.Ng2+		Ke4	(41)	(Kg5)
91.Nh4		Be8	(42)	(Kf4)
92.Ng2		Bc6	(43)	(Bh5)
93.Ne1		Bb5	(44)	(Bb7)
94.Ng2		Bd6	(45)	(Bc6)
95.Nh4		Kf4	(46)	(Bc7, and 11 other equally optimal moves)
96.Ng2+		Kf5	(45)	
97.Ne3+		Kg5	(48)	(Ke4)
98.Ng2	(Nd5)	Bc6	(43)	
99.Ne1		Kg4	(52)	(Kf5)
100.Ke3		Bc5+	(51)	
101.Kd3		Bf2	(50)	
102.Nc2		Kf4	(49)	
103.Na3	(Nd4)	Bh4	(49)	(Be4+)

White	(optimal)	Black	depth	(optimal)
104.Kd4		Be8	(48)	
105.Nc4		Bf7	(51)	(Bf2+)
106.Kc5	(Nd6)	Bf2+	(33)	
107.Kb4		Bd4	(32)	
108.Kb5		Bh5	(47)	(Be8 .)
109.Kc6		Bf3+	(50)	(Ke4)
110.Kc7	(Kd6)	Kf5	(46)	
111.Na5	(Kd6)	Ke6	(44)	
112.Nb7	(Nc6)	Ke7	(45)	(Be5+)
113.Na5		Bf2	(44)	
114.Nc6+		Ke6	(43)	
115.Na5	(Nd8+)	Bg3+	(41)	(Ke4)
116.Kb6		Bf2+	(40)	
117.Kc7				

At this point the game was declared drawn by the '50-move rule', as no pawn had been moved and no capture had taken place for 50 consecutive moves.

We permit ourselves five factual observations and one psychological comment.

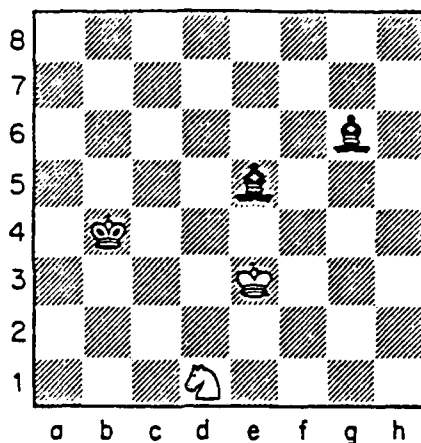
Firstly, it may be seen that Pinter set up the Kling and Horwitz position no fewer than three times, each time in a different corner, namely after his moves 71 (in the b2 area), 90 (in the g2 area) and 112 (in the b7 area).

Secondly, Pinter's knowledge of the Kling and Horwitz position leads him consistently to head for it with unnecessary haste, this accounting for a number of his sub-optimal choices.

Thirdly, the quality of an individual sub-optimal move by either side can be crudely measured (if the opponent's previous and subsequent moves are optimal) by comparing the successive depth numbers. Thus it can be seen that Bronstein's 79...Kc5; increased the optimal depth from 35 to 45, a cruel penalty for not playing the optimal and so similar Kc4; while Pinter's 88.Kf2, reduced the optimal depth from 50 to 41. The great difficulty of this endgame is evident when one tries to give well-grounded reasons for the played moves being inferior to the optimal moves. (See R24.)

Fourthly, an optimal win within the confines of the traditional 50-move rule became possible after White's 70th move, but was lost, never to recur, with Black's 79th.

Fifthly, 36 of White's 50 moves were optimal, while only 26 of Black's



R24. Pinter vs. Bronstein. Position after White's 79th move. Is 79...Kc4; or 79...Kc5; the better move, and why?

49 were optimal. A conjecture is that this is evidence for the endgame being more difficult to play for the side with the bishops.

The psychological comment is that a mistake (as distinct from a crude blunder or oversight) of the kind of Bronstein's 79...Kc5; or Pinter's 88.Kf2, although it leaps to the eye when scanning the depth parentheses to the above game score, is recognized by the player, if at all, only several moves later. The player's general strategy will in all probability have been correct at the highest level, but his ability to calculate in order to reject moves that appear to meet the strategic objectives equally well (which nevertheless fail when countered by an optimal continuation) will be insufficient: one or more vital concepts are missing. Before the creation of the oracle data base no one could have described the feeling for position and depth of calculation needed to play this endgame really well. Now it begins to be possible. The missing concepts are waiting to be formed from data in the data base, and to be verified by reference to the same data base. The idea of *automatic* derivation of concepts or patterns meaningful to human domain specialists is a challenging, a tantalizing, possibility—is its realisation just round the corner or is it remote?

Acknowledgements

The 'BBN' data base was generated and generously made available by Mr Ken Thompson of Bell Laboratories, New York, USA. Dr Alen Shapiro of Intelligent Terminals Limited set up the data base in the Turing Institute, added an interactive interface, and was responsible for the admirably smooth conduct of the computing side of the contest. Professor Donald Michie directed the project. I am grateful to The Royal Society and to the Science and Engineering Research Council of Great Britain for the award of an Industrial Fellowship, and to my employer, IBM United Kingdom Limited, for their contribution and associated secondment.

REFERENCES

Entries entirely within brackets are recognized authorities on chess endgame theory not explicitly referred to in this paper.

Alty, J. I. and Coombs, M. J. (1984) *Expert systems—concepts and examples*. NCC, Manchester.

- [Averbakh, Yu. I. (ed.) (1980–1984) *Shakhmatnye okonchaniya*. In five volumes. Fizkul'tura i Sport, Moscow.]
- Benko, P. (July 1984) Electronic arts. In *Chess life*, pp. 44–5, 58. United States Chess Federation, New Windsor, NY.
- Bouwkamp, C. J. (1967) Catalogue of solutions of the rectangular $3 \times 4 \times 5$ solid pentominoes problem. Technological University Eindhoven and Philips Research Laboratories, Eindhoven.
- [Chéron, A. (1960–1970) *Lehr- und Handbuch der Endspiele*. In four volumes. Siegfried Engelhardt, Berlin.]
- Comay, O. and Roycroft, A. J. (April 1984) Two bishops against knight (continued from EG74). *EG*, no. 75. 249–52.
- [Fine, R. (1941) *Basic chess endings*. David McKay, Philadelphia.]
- [Hooper, D. V. (1970) *A pocket guide to chess endgames*. Bell, London.]
- Kling, J. and Horwitz, D. (1851) *Chess studies, or endings of games*. Skeet, London.
- Kopec, D. and Niblett, T. B. (1980) How hard is the play of the king-rook-king-knight ending? In *Advances in Computer Chess 2* (ed. M. R. B. Clarke), pp. 57–81. Edinburgh University Press, Edinburgh.
- Michie, D. (1961) Trial and error. *Science Survey*, 2, 129–45. Reprinted in *On Machine Intelligence* (2nd edn). Ellis Horwood, Chichester (1986).
- Michie, D. (1986) The superarticulacy phenomenon in the context of software manufacture. *Proc. R. Soc. Lond. A* 405, 185–212.
- Nilsson, Nils, J. (1971). *Problem-solving methods in artificial intelligence*. McGraw-Hill, New York.
- Pachman, L. (1983) *Chess endings for the practical player*. Routledge & Kegan Paul, London.
- Rich, E. (1983) *Artificial intelligence*. McGraw-Hill, New York.
- Roycroft, A. J. (1972) *Test tube chess*. Faber & Faber, London.
- Roycroft, A. J. (1981) *The chess endgame study*. Dover, New York.
- Roycroft, A. J. and Niblett, T. (June 1979) How the GBR class 0103 data base was created. *EG* no. 56, 145–6.
- Shirai, Y. and Tsuji, J. (1982) *Artificial intelligence*. John Wiley, Chichester.
- Ströhlein, T. (1970) *Untersuchungen über Kombinatorische Spiele*. Technische Hochschule, Munich.
- Thompson, K. and Roycroft, A. J. (November 1983) A prophecy fulfilled. *EG* no. 74, 217–20.
- Thompson, K. (May 1986) *C* the programs that generate endgame data bases. *EG* no. 83, 2.

Inductive Acquisition of Chess Strategies

S. H. Muggleton*

Edinburgh University, UK

Abstract

A variation of an algorithm for inducing ' k -contextual' regular language grammars from sample sentences is applied to the construction of expert chess strategies. In a pilot study a small expert system for playing part of the king and two bishops against king and knight endgame (KBBKN) has been automatically constructed using this technique. The generated knowledge-base is directly executable in a MUGOL environment. Although this work is indicative of a new methodology for automatically generating chess-playing strategies from example sequences of play, further work is necessary to show that the technique would be generally applicable to this task.

1. INTRODUCTION

1.1. Computer chess research

In the study of expert system development, Michie has noted that use of chess expertise as a testbed domain is ideal in many respects. The domain is non-trivial though finitely bounded. It has a wealth of recorded expertise going back many centuries which has certainly not yet been fully exercised. Whereas chess specialists have developed a depth of understanding which is at least comparable with the expertise of more lucrative disciplines, expert-level chess players are generally more readily available for consultation.

Early work in programming computers to play chess was concentrated around efficiently implementing Shannon's chess playing strategy [1]. This employs extensive lookahead in order to compute approximations to the best next move. As this failed to produce results comparable with human expert play, recent research has focused on more knowledge-rich approaches. Bratko and Michie [2] described such a knowledge-based system, ALI, based partly on earlier work by Huberman [3]. ALI's *advice*

* Present address: The Turing Institute, George House, Glasgow, UK.

module generated a list of preference-ordered pieces of advice. A separate *search module* used the board-state and advice list to produce a 'forcing tree' which was applied as a strategy for play. As with all solutions in which knowledge must be hand-coded, the knowledge acquisition process becomes a developmental bottleneck.

Quinlan [4] suggested a method of bypassing this bottleneck by using inductive inference. Quinlan's algorithm, ID3, based on Hunt's CLS algorithm [5], was used to build decision trees which classified endgame positions as won, drawn, or lost. A vector of attribute values is used to describe any particular position. This vector together with a class value comprises an example classification. Although the solutions were exhaustively proved correct and ran up to two orders of magnitude faster than commonly used algorithms, they were also completely incomprehensible to chess experts.

In order to circumvent this understandability barrier Shapiro and Niblett [6] introduced the notion of *structured induction*, in which a chess expert is required to decompose hierarchically the endgame classification rules; each subproblem can then be solved inductively. While this approach avoids the problem of incomprehensibility, it unfortunately introduces a new bottleneck of problem structuring.

Paterson [7] has described an attempt to structure automatically the KPK chess endgame domain from example material, using the statistical clustering algorithm CLUSTER. The results, however, have not been very promising, with the machines suggested hierarchy not having any significance to experts. The primary reasons for failure seem to lie in the fact that although the example set is a rich enough source of knowledge to be used for rule construction, additional information is necessary to indicate any higher-level structure.

1.2. Sequence induction

In this paper we describe a new approach to the automatic construction of chess strategies from example material. Note that this differs considerably from the approaches of Quinlan [4], and Shapiro and Niblett [6]. In their case, a diagnostic or classificatory expert system was inductively built using static 'snapshot' descriptions. In ours, we build a procedural or strategic expert system from dynamic 'sequence' examples. Each element of the sequence is a snapshot like example of the ID3 variety. The output of the inductive process is a finite state structure in which each state contains a small number of the snapshot examples. These can in turn be used by ID3-like induction schemes to produce rules or decision-trees for each state. *Thus although we do not produce a hierarchical structure, we achieve the aims of structured induction (i.e. a set of small understandable rules) by using example material which contains additional structural information within each example.*

The basis for these techniques lies in the study of grammatical induction, that is the inference of grammatical structures from example sentences of a language [8]. The grammar produced can be viewed as the control structure of a program which generated the example sentences. Some of the earliest work in this area was done by Biermann and Feldman [9] who devised an algorithm to induce finite state automata from strings of a language. Although their algorithm was capable of finding any regular language given a sufficient example set, the algorithm requires an arbitrary complexity parameter and also has rather low *example efficiency* (i.e. a large number of examples are needed to infer anything). Angluin [10] has described an algorithm which infers only a limited subset of the regular languages. This subset she calls the *k-reversible* languages. By limiting the target result set, Angluin's algorithm achieves example efficiency higher than that of Biermann and Feldman's algorithm.

The author [11] has taken Angluin's algorithm and redesigned it to run with $O(n^2)$ time complexity rather than Angluin's original $O(n^3)$ time. Furthermore, we have discovered an even smaller, but useful subset of the *k-reversible* languages, which we call the *k-contextual* languages. The algorithm for inferring members of the *k-contextual* languages is again more example efficient than even Angluin's, to the extent that sensible inference is possible from samples containing only a single example (all other methods in the literature [9, 10, 12, 13] presuppose more than a single example). The *k-contextual* algorithm has $O(n)$ time complexity.

2. THE *k*-CONTEXTUAL ALGORITHM

Grammatical induction techniques use exemplary sentences to form generalized grammatical descriptions which are at least compatible with the original example material. Examples can come in two different forms, positive examples and negative examples. Positive examples are members of the target grammar, while negative examples are not. If only positive examples are used then the inductive process must use well-defined constraints on permissible solutions in order to avoid over-generalization. Alternatively, these constraints can be provided by the use of negative examples, in which case, any generated descriptions must hold for all examples that are positive and for none that are negative.

The *k-contextual* algorithm used for the experiments described here requires only positive examples. The necessary constraint on solutions is that the finite state acceptor produced be equivalent to the minimum-size *k-contextual* language containing the positive examples [11]. A regular language L is *k-contextual* if and only if whenever two not necessarily distinct strings u_1vw_1 and u_2vw_2 are elements of L and v has length k , then u_1vw_2 and u_2vw_1 are also elements of L . For normal grammatical

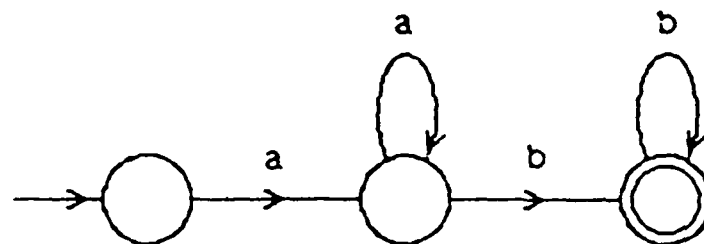


Figure 1. An hypothesized finite state acceptor for the grammatical sample $\{aabb\}$.

structures, k is a parameter which must be supplied to the algorithm and can be thought of as a complexity measure for the solution. Generally the smaller the value of k the larger the accepted language. However, when dealing with sequences of ID3-like examples, we can use the semantic content provided by the situational vector as an additional constraint mechanism, and thus circumvent the need for supplying the algorithm with the arbitrary measure required by all similar algorithms in the literature [9, 10, 12, 13]. This is achieved by first looking for $k=0$ solutions, and then if a 'clash' (non-determinism) is produced in any of the states of the solution, the value of k is incremented. This process is repeated until either a deterministic solution is produced and the algorithm returns successfully, or it reaches a maximum possible value equal to the maximum length of example string, and returns with failure.

Figure 1 portrays an example of the application of grammatical induction to a set of example sentences (clearly the set contains only one member, i.e. $aabb$). The k -contextual algorithm hypothesis represents the language a^+b^+ .

Situations in which sequence induction can be employed are many and varied [4]. If we understand well what the properties of the algorithm being used are, often we can take advantage of various presentation and solution constraints for different scenarios. Elsewhere [11] several such properties are theoremtically described and proved. The most important such property is what Gold [15] calls *identification in the limit*. Let a grammatical induction algorithm I make a hypothesis of a language L_i after each example sentence u_i presented by a complete, arbitrarily ordered enumeration of such examples. I is said to identify the target language L in the limit if and only if there exists some finite natural number n such that I hypothesises the correct language $L_n = L$ following the example u_n and does not subsequently change its guess. The k -contextual algorithm used here has been used [11] to identify k -contextual languages in the limit.

3. THE PROBLEM—KBBKN

Programming strategies for chess endgames is a notoriously difficult task. Zuidema [16] commenting on two Algol 60 programs written for the KRK

endgame illustrates the difficulties by noting that 'A small improvement entails a great deal of expense in programming effort and program length. The new rules will have their exceptions too.'

In a project being carried out at the Turing Institute, the extremely complex chess endgame KBBKN is being studied with the aid of the world-class chess endgame specialist John Roycroft. Even this chess authority claims to be out of his depth. In the only definitive study of KBBKN, written in 1851, Horwitz and Kling [17] declared that with White-to-move (WTM), the game is drawn in all but trivial cases. For over a century this claim remained uncontested, until in 1983 Thomson [18] revealed by exhaustive computation that almost all positions are forced wins for White, with a maximum length win of 66 moves being obtainable from 32 different positions [18, 19].

The Turing Institute study involves two phases. In the first, Roycroft has studied the domain intensely with the aim of developing a full descriptive matrix. It is in this first phase that the author has carried out the evaluation of sequence induction as a knowledge acquisition tool. In the second phase it is intended that Roycroft's descriptions be matched against Thomson's exhaustive database for KBBKN.

Roycroft's first task was to select a sub-strategy within the KBBKN domain of an appropriate size and complexity for the application of sequence induction. The choice fell on the first section of the exceptional 66-move forced win for White.

3.1. Initial position

Play commences from the position shown in Figure 2.

Taking symmetry and slightly altered starting positions into account, this position is equivalent, in terms of the number of moves to a forced win, to several other similar positions. As this equivalence can be taken into account by the choice of terms in the devised expert system, we will ignore this extra dimension to the problem.

3.2. Goal position

The aim of White in this sub-strategy is to liberate wB(light) from the corner in no more than 12 moves. In order to achieve this it is necessary

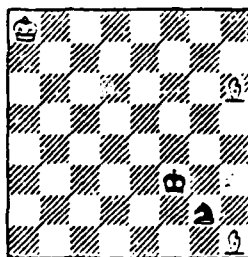


Figure 2. The initial position, WTM.

that

(A) wB(dark) prevents bK from attacking and capturing wBh1. This is illustrated in Figures 3-5.

(B) wK moves to support the attack of wBh1 on bNg2 (Figure 6).

Play achieving (A) is trivially described and encoded. However, attaining (B) is complicated considerably by White's choice of delaying tactics, employed to impede wK approaching h3. It was for this second goal that we use sequence induction to capture Roycroft's description.

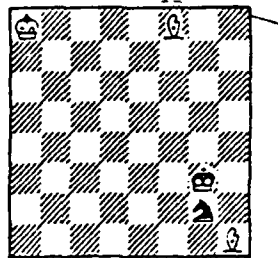


Figure 3. wB(light) prepares to prevent wK from moving to h2, WTM.

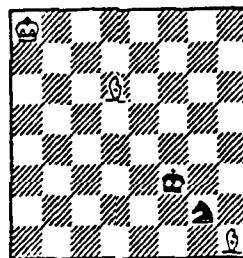


Figure 4. bK retreats after being checked by wB(light), WTM.

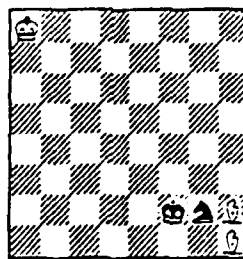


Figure 5. wB(light) takes up fortified position, WTM.

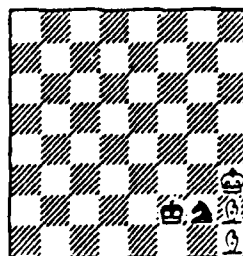


Figure 6. The goal of liberating wB(dark) is achieved, bN is forced to retreat, BTM.

3.3. Attributes and actions

Roycroft was asked to give an exposition of play which included a set of sequences of moves together with a running commentary displaying points of interest. From this the author extracted four positional attributes (based on Roycroft's use of adjectival phrases), four actions taken by White (corresponding to verb phrases), and six sequences of play. The attributes were as follows

- (B1) Is White free to take bN? {y/n}
- (B2) Is wK on the same diagonal as the release position (h3)? {y/n}
- (B3) Can wBh1 (dark) move? {y/n}
- (B4) Is the direct diagonal position closest to the release position covered? {y/n}

The actions were

- (Ba) wK approaches release position (h3) by moving along rank or file.
- (Bb) wK moves to non-check position closest to release position on direct diagonal.
- (Bc) wB(light) moves out of corner along its diagonal..
- (Bd) White takes bN.

Note that each action at this level represents a single move. However, the entire automaton to be derived represents a unit action involving several moves. Thus we might, if necessary, have a hierarchy of such actions and attributes, similar to that described by Shapiro and Niblett [6] for classification (see discussion).

3.4. The solution

The sequences used are reproduced in Appendix A. These were presented to a PROLOG-coded version of the *k*-contextual algorithm, the output being translated into a suitable form for further m3-like induction and run-time testing in the MUGOL environment. Sequences were added by stepwise-refinement, the result being tested after the addition of each sequence. Very early in this process, the *k*-value for the solution rose from 0 to 1, at which level it remained during the rest of development. Also, the number of states in the solution grew rapidly at first to reach a steady value of 5, at which it too stayed fixed. Altogether this process displayed a good incremental nature.

The first six sequences represent White's response to various well-executed tactics played by Black. These were derived directly from Roycroft's description. Having by this stage generated a playing strategy that dealt adequately with more than Roycroft's described positions (the *k*-contextual algorithm successfully generalized solutions to a larger number of positions than those originally described) the automaton was presented and explained to Roycroft. Roycroft noted that the set of positions at which the white king can be delayed by Black was the most

complex to describe. *Significantly, the state which described just these positions contained the most ID3-vectors. Thus the structure automatically imposed on the solution had a clear significance to the expert.*

As yet, with only six sequences, the solution was not able to cope with bad play by Black. An additional seven sequences were added to deal with such play. The resulting k -contextual automaton is given in Appendix B in a form which can be directly translated into a MUGOL [20] induction file. Appendix C demonstrates the transformation carried out by ID3-like induction to produce a runnable MUGOL expert system. Note that all decision-trees in the solution have the form of HSL [21] decision-trees.

4. DISCUSSION

In this paper we have indicated the feasibility of using sequence induction to construct expert-level chess strategies for endgame play. A great deal of further work is necessary to show that:

- (i) *optimal* playing strategies can be produced using the technique outlined here;
- (ii) solutions generally can be found from *any chosen section* in the KBBKN domain; and
- (iii) generated solutions generally are, or can be made to be, *conceptually transparent* to the expert who provided the example material.

For the chosen subgame described here, the methodology used was found by the expert to be natural in terms of the example presentation requirements, as chess players are quite at home with describing play in terms of example move sequences. Furthermore, the bottleneck of structuring was eased, though not completely removed by the use of sequence induction. Whereas other attempts at automatic structuring have led to solutions which are not acceptable to experts, results produced by sequence induction were found to be intuitively correct by the endgame specialist John Roycroft.

In Section 3.3 we noted that as the induced strategy represents a unit action, it might be found necessary to form a hierarchy of such actions in order to create an extensive strategy. Therefore, it might be argued that our automatic structuring aid has gained us no ground, as it may still be necessary to do further manual structuring. We do not claim to have a complete answer to the structuring problem. However, Shapiro [22] in his structured solution of KPa7KR used an average of six examples at each inductive stage in order to produce readable decision-trees. We have used 13 example sequences each containing an average of four ID3-like sequences to produce a semi-structured solution in which each state's rule is derived from an average of only three examples. The example material

used here thus consists of approximately $13 \times 4 = 52$ situation/action pairs. Despite the fact that the quantity of example material used to structure this level of problem is an order of magnitude larger than that used by Shapiro, the generated solution contains a small number of easily understandable decision trees.

The k -contextual induction algorithm used was found to display good incremental behaviour. This is true in general for this algorithm, which has been proved to identify k -contextual solutions in the limit.

On the negative side, we have not developed a form of explanation which deals satisfactorily with sequence execution. It is hoped that by continued research, John Roycroft may be able to suggest a more natural form of explanation in line with that used by chess players to describe sequences of play. Furthermore, since it was necessary to hand-translate ID3-like example material from the PROLOG output form of the k -contextual algorithm, it was clear that a better interface to the MUGOL environment is needed.

Acknowledgements

This research was carried out while the author was studying for a Ph.D. at Edinburgh University sponsored by the SERC. Thanks are due to the Turing Institute and Edinburgh University AI department for resource and facilities.

We would also like to thank Professor Michie and John Roycroft for their help and guidance, and Thirza A. Castello-Cortes for helping to get it written on time.

REFERENCES

1. Shannon, C. E. (1950) Programming a computer for playing chess. *Phil. Mag.* **41**, 256-275.
2. Bratko, I. and Michie, D. (1980) A representation of pattern-knowledge in chess endgames. *Advances in computer chess 2* (ed. M. R. B. Clarke) pp. 31-56. Edinburgh University Press, Edinburgh.
3. Huberman, B. J. (1968) A program to play chess end-games. Technical report no. CS 106, Computer Science Department, Stanford University.
4. Quinlan, J. R. (1982) Semi-autonomous acquisition of pattern-based knowledge, *Introductory readings in expert systems*, pp. 192-207. Gordon & Breach, New York.
5. Hunt, E. B., Marin, J., and Stone, P. (1966) *Experiments in induction*. Academic Press, New York.
6. Shapiro, A. and Niblett, T. (1982) Automatic induction of classification rules for a chess endgame. *Advances in computer chess 3* (ed. M. R. B. Clarke) pp. 73-92. Pergamon Press, Oxford.
7. Paterson, A. (1983) An attempt to use CLUSTER to synthesise humanly intelligible subproblems for the KPK chess endgame. University of Illinois series (UIUCDCS-R-83-1156).
8. Muggleton, S. H. (1984) Induction of regular languages from positive examples. *MINews* **5**, 41-59, Turing Institute, Glasgow.
9. Biermann, A. W. and Feldman, J. A. (1972) On the synthesis of finite-state machines from samples of their behaviour. *IEEE Transactions on Computers* **C21**, 592-597.
10. Angluin, D. (1982) Inference of reversible languages. *J. Association for Computing Machines* **29**, 741-756.

INDUCTIVE ACQUISITION OF CHESS STRATEGIES

11. Muggleton, S. (1986) Inductive acquisition of expert knowledge. Ph.D. thesis, University of Edinburgh.
12. Levine, B. (1982) The use of tree derivatives and a sample support parameter for inferring tree systems. *IEEE Transactions of Pattern Analysis and Machine Intelligence* PAMI4, 25-34.
13. Miclet, L. (1980) Regular inference with a tail clustering method. *IEEE Transactions on Systems, Man, Cybernetics* SMC10, 737-743.
14. Muggleton, S. (1985) Some experiments with grammatical induction. *MINews* 7, 60-74.
15. Gold, E. M. (1967) Language identification in the limit. *Inf. Control* 10, 427-474.
16. Zuidema, C. (1974) Chess, how to program the exceptions? *Afdeling informatica IW21/74 Mathematisch Centrum, Amsterdam*.
17. Horwitz and Kling (1851) *Chess studies*. C. J. Skeet, London.
18. Thompson, K. (1985) Letter to J. Roycroft. *EG* (April 1985).
19. Roycroft J. (1983) A prophesy fulfilled. *EG* (Nov. 1983).
20. Michie, D., Muggleton, S., Riese, C., and Zubrick, S. (1984) RuleMaster: a second-generation knowledge-engineering facility. *Proc. First Conf. on Artificial Intelligence Applications* (1984).
21. Michie, D. (1984) Quality control of induced rule-based programs. *The fifth generation*, CGS Institute notes.
22. Shapiro, A. D. The role of structured induction in expert systems. Ph.D. thesis, University of Edinburgh.

APPENDIX A—EXAMPLE MOVE SEQUENCES

Actions

(Ba) wK approaches release position (e.g. h3) by moving along rank or file.

(Bb) wK moves to non-check position on direct diagonal which is closest to release position.

(Bc) wB(light) moves out to corner along its diagonal.

(Bd) white takes bN.

Attributes

(B1) White free to take bN.

(B2) wK on the same diagonal as release position.

(B3) wBh1 can move.

(B4) (wK on direct diagonal) and (direct diagonal position closest to release position is covered).

Sequence 1. Starts from wKa8 and bN does delaying check.

B1	B2	B3	B4	Action	Position	Move
n	n	n	n	Ba	wKa8 wBh1 wBh2 bKf3 bNg2	wKb8
n	n	n	n	Ba	wKb8 wBh1 wBh2 bKf2 bNg2	wKc8
n	y	n	n	Bb	wKc8 wBh1 wBh2 bKf3 bNg2	wKd7
n	y	n	n	Bb	wKd7 wBh1 wBh2 bKf2 bNg2	wKe6
n	y	n	n	Bb	wKe6 wBh1 wBh2 bKf1 bNg2	wKf5
n	y	n	n	Ba	wKf5 wBh1 wBh2 bKf1 bNe3	wKg5
n	n	n	n	Bb	wKg5 wBh1 wBh2 bKf1 bNg2	wKg4
n	y	n	n	Bb	wKg4 wBh1 wBh2 bKf2 bNg2	wKh3
n	-	y	n	Bc	wKh3 wBh1 wBh2 bKf1 bNf3	wBa8

The '-' in the last line allows the algorithm to generalize to the case in which bN releases wB(light).

Sequence 2. Starts from wKb7 and bN does delaying check.

B1	B2	B3	B4	Action	Position	Move
n	n	n	n	Ba	wKb7 wBh1 wBh2 bKf2 bNg2	wKc7
n	n	n	n	Ba	wKc7 wBh1 wBh2 bKf3 bNg2	wKd7
n	y	n	n	Bb	wKd7 wBh1 wBh2 bKf2 bNg2	wKe6
n	y	n	n	Bb	wKe6 wBh1 wBh2 bKf1 bNg2	wKf5
n	y	n	n	Ba	wKf5 wBh1 wBh2 bKf1 bNe3	wKg5
n	n	n	n	Bb	wKg5 wBh1 wBh2 bKf1 bNg2	wKg4
n	y	n	n	Bb	wKg4 wBh1 wBh2 bKf2 bNg2	wKg3
n	-	y	n	Bc	wKh3 wBh1 wBh2 bKf1 bNf3	wBa8

Sequence 3. Starts from wKb8 and bN does delaying check.

B1	B2	B3	B4	Action	Position	Move
n	n	n	n	Ba	wKb8 wBh1 wBh2 bKf2 bNg2	wKc8
n	y	n	n	Bb	wKc8 wBh1 wBh2 bKf3 bNg2	wKd7
n	y	n	n	Bb	wKd7 wBh1 wBh2 bKf2 bNg2	wKe6
n	y	n	n	Bb	wKe6 wBh1 wBh2 bKf1 bNg2	wKf5
n	y	n	n	Ba	wKf5 wBh1 wBh2 bKf1 bNe3	wKg5
n	n	n	n	Bb	wKg5 wBh1 wBh2 bKf1 bNg2	wKg4
n	y	n	n	Bb	wKg4 wBh1 wBh2 bKf2 bNg2	wKg3
n	-	y	n	Bc	wKh3 wBh1 wBh2 bKf1 bNf3	wBa8

INDUCTIVE ACQUISITION OF CHESS STRATEGIES

Sequence 4. Starts with wKa8 and bN does not do delaying check.

B1	B2	B3	B4	Action	Position	Move
n	n	n	n	Ba	Wka8 wBh1 wBh2 bKf3 bNg2	wKb8
n	n	n	n	Ba	wKb8 wBh1 wBh2 bKf2 bNg2	wKc8
n	y	n	n	Bb	wKc8 wBh1 wBh2 bKf3 bNg2	wKd7
n	y	n	n	Bb	wKd7 wBh1 wBh2 bKf2 bNg2	wKe6
n	y	n	n	Bb	wKe6 wBh1 wBh2 bKf1 bNg2	wKf5
n	y	n	n	Bb	wKf5 wBh1 wBh2 bKf2 bNg2	wKg4
n	y	n	n	Bb	wKg4 wBh1 wBh2 bKf1 bNg2	wKh3
n	-	y	n	Bc	wKh3 wBh1 wBh2 bKf2 bNf3	wBa8

Sequence 5. Starts with wKg4.

B1	B2	B3	B4	Action	Position	Move
n	y	n	n	Bb	wKg4 wBh1 wBh2 bKf1 bNg2	wKh3
n	-	y	n	Bc	wKh3 wBh1 wBh2 bKf2 bNf3	wBa8

Sequence 6. Starts with wKh3.

B1	B2	B3	B4	Action	Position	Move
n	-	y	n	Bc	wKh3 wBh1 wBh2 bKf2 bNf3	wBa8

Black plays badly

Sequence 7. Starts with wKa8 after bK has left bN undefended (en prise).

B1	B2	B3	B4	Action	Position	Move
y	-	n	n	Bd	wKa8 wBh1 wBh2 bKe2 bNg2	wB x N!

Sequence 8. Starts with wKa8 and bK leaves bN as first move.

B1	B2	B3	B4	Action	Position	Move
n	n	n	n	Ba	wKa8 wBh1 wBh2 bKf3 bNg2	Wkb8
y	-	n	n	Bd	wKb8 wBh1 wBh2 bKe3 bNg2	wB x N!

Sequence 9. Starts with wKg4 and bK leaves bN as first move.

B1	B2	B3	B4	Action	Position	Move
n	y	n	n	Bb	wKg4 wBh1 wBh2 bKf1 bNg2	wKh3
y	-	n	n	Bd	wKh3 wBh1 wBh2 bKe1 bNg2	wB × N!

Sequence 10. Starts with wKb8, bN does not do delaying check but allows the release of wB.

B1	B2	B3	B4	Action	Position	Move
n	n	n	n	Ba	wKb8 wBh1 wBh2 bKf2 bNg2	wKc8
n	-	y	n	Bc	wKc8 wBh1 wBh2 bKf2 bNd1	wBc6

Sequence 11. Starts with wKe6, bN does delaying check and then allows the release of wB.

B1	B2	B3	B4	Action	Position	Move
n	y	n	n	Bb	wKe6 wBh1 wBh2 bKf1 bNg2	wKf5
n	y	n	n	Ba	wKf5 wBh1 wBh2 bKf1 bNe3	wKg5
n	-	y	n	Bc	wKg5 wBh1 wBh2 bKf1 bNd1	wBc6

Sequence 12. Starts with wKe6, bN does delaying check and then allows itself to be taken (by moving to g4).

B1	B2	B3	B4	Action	Position	Move
n	y	n	n	Bb	wKe6 wBh1 wBh2 bKf1 bNg2	wKf5
n	y	n	n	Ba	wKf5 wBh1 wBh2 bKf1 bNe3	wKg5
y	-	n	n	Bd	wKg5 wBh1 wBh2 bKf1 bNg4	wKg4!

Sequence 13. Starts with wKd7, bN checks allowing itself to be taken by wB(dark).

B1	B2	B3	B4	Action	Position	Move
n	y	n	n	Bb	wKd7 wBh1 wBh2 bKf2 bNg2	wKe6
y	y	y	n	Bd	wKe6 wBh1 wBh2 bKf2 bNf4	B × N

APPENDIX B—RESULT OF SEQUENCE INDUCTION

Actions

(Ba) wK approaches release position (e.g. h3) by moving along rank or file.

(Bb) wK moves to non-check position on direct diagonal which is closest to release position.

(Bc) wB(light) moves out of corner along its diagonal.

(Bd) White takes bN.

Attributes

(B1) White free to take bN.

(B2) wK on the same diagonal as release position.

(B3) wBh1 can move.

(B4) (wK on direct diagonal) and (direct diagonal position closest to release position is covered)

B1 B2 B3 B4 (Action, Next State)

State 0

n - y n \Rightarrow (Bc, GOAL)

n n n n \Rightarrow (Ba, 1)

n y n n \Rightarrow (Bb, 2)

y - n n \Rightarrow (Bd, GOAL)

State 1

n - y n \Rightarrow (Bc, GOAL)

n n n n \Rightarrow (Ba, 1)

n y n n \Rightarrow (Bb, 2)

y - n n \Rightarrow (Bd, GOAL)

State 2

n - y n \Rightarrow (Bc, GOAL)

n y n n \Rightarrow (Bb, 2)

n y n y \Rightarrow (Ba, 3)

y - n n \Rightarrow (Bd, GOAL)

y y y n \Rightarrow (Bd, GOAL)

State 3

n - y n \Rightarrow (Bc, GOAL)

n n n n \Rightarrow (Bb, 4)

y - n n \Rightarrow (Bd, GOAL)

State 4

n y n n \Rightarrow (Bb, 2)

APPENDIX C—AUTOMATA AFTER 103-LIKE INDUCTION

Actions

(Ba) wK approaches release position (e.g. h3) by moving along rank or file.

(Bb) wK moves to non-check position on direct diagonal which is closest to release position.

(Bc) wB(light) moves out of corner along its diagonal.

(Bd) White takes bN.

Attributes

(B1) White free to take bN.

(B2) wK on the same diagonal as release position.

(B3) wBh1 can move

(B4) (wK on direct diagonal) and (direct diagonal position closest to release position is covered).

State 0

[B1]

y: \Rightarrow (Bd, GOAL)

n: [B3]

y: \Rightarrow (Bc, GOAL)

n: [B2]

y: \Rightarrow (Bb, 2)

n: \Rightarrow (Ba, 1)

State 1

[B1]

y: \Rightarrow (Bd, GOAL)

n: [B3]

y: \Rightarrow (Bc, GOAL)

n: [B2]

y: \Rightarrow (Bb, 2)

n: \Rightarrow (Ba, 1)

State 2

[B1]

y: \Rightarrow (Bd, GOAL)

n: [B3]

y: \Rightarrow (Bc, GOAL)

n: [B4]

y: \Rightarrow (Ba, 3)

n: \Rightarrow (Bb, 2)

State 3

[B1]

y: \Rightarrow (Bd, GOAL)

n: [B4]

y: \Rightarrow (Bc, GOAL)

n: \Rightarrow (Bb, 4)

State 4

(2)

Inverting the resolution principle.

Stephen Muggleton*
Turing Institute
Glasgow

September 1987

Abstract. In this paper we describe the current status of an ongoing research project investigating a novel form of Machine Learning in which the learner's vocabulary is enriched by the machine suggesting useful new descriptive terms for the user to accept or reject. An algorithm called Duce has been shown to be effective along these lines in developing and extending propositional theories within a chess endgame domain and a diagnostic domain of neuro-psychology. By showing that Duce's transformational operators are based on reversing the steps of a resolution proof we show that Duce's learning method is sufficient for learning any propositional theory.

1 Introduction

Duce [4] is an algorithm which produces hierarchical concept descriptions from large numbers of examples. Whereas the ID [7] and AQ [3] families of inductive algorithms require all necessary attributes to be provided before learning can take place, Duce develops new attributes by incrementally building them from existing ones, testing each against the user for comprehensibility. Duce uses a set of transformations of propositional Horn clauses which successively compress the example material on the basis of generalisations and the additions of new terms. In the following description of three of the six Duce operators lower-case Greek letters stand for conjunctions of propositional symbols.

1. Intra-construction. This is the distributive law of Boolean equations. We take a set of rules such as

$$\begin{aligned}h_1 &\leftarrow \alpha\beta \\ h_1 &\leftarrow \alpha\gamma\end{aligned}$$

and replace them with the rules

$$\begin{aligned}h_1 &\leftarrow \alpha h_3 \\ h_3 &\leftarrow \beta \\ h_3 &\leftarrow \gamma\end{aligned}$$

The user either names the new concept h_3 or rejects it.

*This paper describes work which was funded in part by the British Government's Alvey Logic Database Demonstrator. Research facilities were provided by the Turing Institute, Glasgow, UK and Interact R&D Corporation, Victoria, BC, Canada.

2. **Absorption.** This operator is due to Sammut and Banerji [10]. Given a set of rules, the body of one of which is completely contained within the bodies of the others, such as

$$\begin{aligned}h_1 &\leftarrow \alpha\beta \\h_2 &\leftarrow \alpha\end{aligned}$$

one can hypothesise

$$\begin{aligned}h_1 &\leftarrow h_2\beta \\h_2 &\leftarrow \alpha\end{aligned}$$

The user can either accept this generalisation or reject it.

3. **Identification.** This operator has preconditions which are stronger than those of intra-construction. A set of rules which all have the same head, the body of at least one of which contains exactly one symbol not found within the other rules, such as

$$\begin{aligned}h_1 &\leftarrow \alpha\beta \\h_1 &\leftarrow \alpha h_2\end{aligned}$$

can be replaced by

$$\begin{aligned}h_1 &\leftarrow \alpha h_2 \\h_2 &\leftarrow \beta\end{aligned}$$

Again the user can either accept this generalisation or reject it.

Duce uses the compaction of the rulebase produced by each of the six operators to guide the search for the next operator to apply. In [4] we give the set of characteristic formulae, one for each operator which predict the exact *symbol reduction* produced by each operator, the number of *symbols* in a rule being equal to the rule-body length plus one for the rule-head. Since Duce only applies operators which give a positive symbol reduction it can be easily shown that the algorithm terminates after a finite number of operator applications.

2 Application domains.

2.1 KPa7KR application.

The first large-scale test of Duce's capabilities[4], was an attempt to automatically reconstruct Shapiro and Kopec's expert system [11] for deciding whether positions within the endgame of King-and-Pawn-on-a7 v. King-and-Rook were won for white or not. A set of 3196 examples were used, and Duce's questions were answered by the chess endgame specialists Ivan Bratko and Tim Niblett. The result was a comprehensible restructuring of the domain, topologically similar to Shapiro and Kopec's original structure, though an order of magnitude more bulky.

2.2 Neuropsychology application.

A second, and previously undescribed structuring experiment was carried out by the author using Duce at Interact Corporation, Canada. In this Duce was used to construct a problem decomposition for deciding on dysfunction of the left parietal brain area of children with learning disabilities. The input to the algorithm consisted of 227 diagnosed cases. Each case contained the results of a battery of approximately 100 binary-valued clinical tests. Each case was marked with a diagnosis of normal/abnormal left parietal lobe by the resident clinical neuro-psychologist, Dr. Russell. Using these cases Duce carried out an interactive session in which Russell was asked to answer a total of 53 questions. During and subsequent to the construction of the rulebase, a set of 48 independent cases were used to test the performance of the new rule-set. Since the cases and generated rules were inherently noisy, a majority-vote mechanism was used for rule evaluation. After all 53 questions had been answered, 43 of the 48 test cases agreed with Russell's diagnosis, i.e. 90% agreement. In contrast, an existing expert system developed by Russell had only a 63% agreement rate with Russell's diagnoses over the same test data. While Duce's structured rulebase took 2-3 person-days to build and verify, the equivalent part of the hand-built expert system is conservatively estimated to have taken 2-3 person-months to generate, improve and verify.

In parallel with the supervised construction of the Duce rulebase, Duce was run on the same cases in unsupervised mode. In this mode, all generalisation questions were answered affirmatively and all new concepts were arbitrarily named. Performance with unsupervised learning stabilised after 27 questions to a level of 25% agreement with Russell's diagnoses of the same test cases.

Unlike the endgame experiment in which an exhaustive example set was used, the neuro-psychological example set was relatively sparse. As a consequence, whereas no rejections were necessary in the case of the chess experiment, an average of 10 rejections were required per acceptance with the neuropsychological data. This seems to indicate the need for expert supervision of Duce where sparse data is involved, and explains the dramatic difference in verification results between the supervised and unsupervised data.

The structure of the rulebase created by Russell working with Duce is shown in Appendix A. This hierarchical structure contains groups of rules associated with each node of the network. The sub-types implied by this hierarchy were, according to Russell, "clinically significant", and relate directly to neuropsychological sub-types based on Wide-range-achievement-test (WRAT) results in arithmetic, reading and spelling.

3 Theory.

Duce has shown a considerable amount of success within the application domains described in the previous section. However, there are a number of questions concerning the methodology employed within Duce which are of interest both from a theoretical and a practical viewpoint.

- **Completeness of operators.** Six operators are used by Duce to carry out generalisations and introduce new terms. How complete are these? Are they sufficient to learn any arbitrary set of propositional clauses given enough examples? (see section 3.2)
- **Search.** Duce presently searches through the set of conjunctions of predicate symbols to find which operator to apply next. For this reason Duce can be myopic in its

choice of new terms. The discrepancy in complexity between Duce's solution and human solutions (see section 2.1) seems to indicate that this problem can be quite severe. New methods of searching for operators are required. (see section 3.3)

- **Extensions to first-order representation.** Bain [2] has described a failed attempt to use Duce to learn a simple chess definition of position legality from only positional attributes. Although the definition can be simply described in first-order predicate calculus the cumbersomeness of Bain's description is not eased by adding extra propositional attributes. This gives incentive to an investigation into extending the Duce approach to deal with first-order predicate calculus. (see [5])

3.1 Inverting resolution.

Although it is apparent to many researchers in Machine Learning that there is a strong relationship between deductive theorem-proving mechanisms and inductive inference, this idea has rarely been investigated to any greater depth than to notice that the idea of *logical subsumption* or *logical implication* are central to both. One exception to this is Plotkin [6], who investigated the idea that

just as unification was fundamental to deduction, so might a converse be of use in induction.

From this idea Plotkin went on to develop the concept of *least general generalisation*, or *anti-unification* of literals and clauses.

Unification is a basic idea within Robinson's [9] theory of resolution. Another important concept within this theory is that of the resolution tautology, or rule of inference. As Plotkin [6] notes

It is interesting that ... the similarity between induction and deduction breaks down ... [with anti-unification]. What is useful is not a concept of unification of two clauses, but the deduction principle called resolution.

We now show that the analogy between deduction and induction can be extended fruitfully, and that in fact the operators used by Duce are merely the inverse of resolution. In a later section, this fact will lead us to a proof of the sufficiency of the Duce operators.

In this section we limit our discussion of resolution to binary resolution of propositional Horn clauses. However, in [5] we extend this analysis to deal with first-order representations. Let C_1 and C_2 be the two clauses

$$\begin{aligned} C_1 &= (h_1 \leftarrow \alpha h_2) \\ C_2 &= (h_2 \leftarrow \beta) \end{aligned}$$

We write the resolvent or resolved product of C_1 and C_2 as

$$C = C_1 \cdot C_2 = (h_1 \leftarrow \alpha\beta)$$

We now define the resolved quotient as follows.

$$C_1 = C/C_2$$

Alternatively, the author calls C_1 the *identificant* of C and C_2 . Similarly

$$C_2 = C/C_1$$

Again we call C_2 the *absorbant* of C and C_1 . Note that in both cases, the resolved quotient is unique for propositions. It is now straightforward to define the absorption and identification operators described informally in section 1.

Definition 1 Given a propositional Horn clause program $P \supseteq \{C, C_1\}$, the absorption operator, *Abs*, transforms P to $P' = (P - \{C\}) \cup \{C/C_1\}$.

Definition 2 Given a propositional Horn clause program $P \supseteq \{C, C_2\}$, the identification operator, *Ident*, transforms P to $P' = (P - \{C\}) \cup \{C/C_2\}$.

We can also define the inverse of both of these operators uniquely.

Definition 3 Given a propositional Horn clause program $P \supseteq \{C_1, C_2\}$, the inverse absorption operator, Abs^{-1} , transforms P to $P' = (P - \{C_2\}) \cup \{C_1 \cdot C_2\}$.

Definition 4 Given a propositional Horn clause program $P \supseteq \{C_1, C_2\}$, the inverse identification operator, Ident^{-1} , transforms P to $P' = (P - \{C_1\}) \cup \{C_1 \cdot C_2\}$.

We now give a formal definition of the intra-construction operator of section 1 and its inverse. Let $A = (h_3 \leftarrow \gamma h_4)$, $BB = \{B_1, \dots, B_n\} = \{(h_4 \leftarrow \delta_1), \dots, (h_4 \leftarrow \delta_n)\}$, $CC = \{C_1, \dots, C_n\} = \{(A \cdot B_1), \dots, (A \cdot B_n)\} = \{(h_3 \leftarrow \gamma \delta_1), \dots, (h_3 \leftarrow \gamma \delta_n)\}$.

Definition 5 Given a propositional Horn clause program $P \supseteq CC$, the intra-construction operator, *Intra*, transforms P to $P' = (P - CC) \cup \{A\} \cup BB$.

Definition 6 Given a propositional Horn clause program $P \supseteq (\{A\} \cup BB)$, the inverse intra-construction operator, Intra^{-1} , transforms P to $P' = (P - (\{A\} \cup CC)) \cup BB$.

Lemma 1 If the program transformation $P \rightarrow P'$ is carried out by either Abs^{-1} , Ident^{-1} or Intra^{-1} then P subsumes P' .

Proof. Follows from the fact that all these operators replace more general clauses with more specific ones. \in

The reader may wonder how A and BB are constructed in the definition of *Intra*. As a special case of Plotkin's [6] *least general generalisation (lgg)* of clauses, we say that γ is the *lgg* of the bodies of clauses within CC ($\text{bodies}(CC)$) if and only if γ is the common intersection of propositional symbols of $\text{bodies}(CC)$. Given γ and a new predicate symbol h_3 , it is straightforward to construct A and BB . In fact, while *Abs* and *Ident* represent the only two ways in which a single resolution step can be reversed, *Intra* is one of a number of ways in which the effects of multiple resolution steps can be reversed. It is only through reversal of multiple resolution steps that the introduction of new predicate symbols becomes possible.

3.2 Completeness of Duce operators.

In order to ensure that the success of the Duce applications described in section 2 was not due to some peculiar property of the domains involved we need to show that the operators used by Duce are sufficient to learn any arbitrary set of propositional clauses. Clearly we need to specify some restriction on the allowable forms of examples used, otherwise Duce could merely be presented with any desired solution as its input. Let P be an arbitrary target propositional Horn clause program. The vocabulary used in P ($vocab(P)$) is then simply the set of propositional symbols in P . The primitive vocabulary of P ($prim(P)$) is the set of symbols not defined in terms of other predicate symbols. Thus $prim(P) = vocab(P) - heads(P)$, where $heads(P)$ is the set of clause heads of clauses within P . Now let E be a set of example propositions from which P can be learned by Duce. A reasonable restriction on allowable forms of examples used would seem to be that

1. P subsumes E , i.e. any statement which can be derived from E can also be derived from P ,
2. each clause body in $bodies(E)$ is composed only of predicate symbols from $prim(P)$ and
3. the vocabulary of P is an extension of that of E , i.e. $vocab(P) - vocab(E) \neq \{\}$.

If these conditions are met then we will say that E is a *legitimate* example set of P . First we define the abstract algorithm $Duce_{(Abs, Intra)}$ which is a non-deterministic version of the Duce algorithm limited to using only the *Abs* and *Intra* operators. In the following an inverse derivation $E \rightarrow P_1 \rightarrow \dots \rightarrow P_n$ is a mixed sequence of absorption and intra-construction transformations of the example set E into the propositional Horn clause program $P = P_n$.

Definition 7 The algorithm $Duce_{(Abs, Intra)}(E)$ returns a set of possible Horn clause programs $H = \{P : P \text{ is an inverse derivation of } E\}$.

We can see H as being the hypothesis space of an algorithm which returns a single hypothesis. Angluin [1] introduced the notion of a *characteristic sample* set of examples for language L as being a set of examples which are sufficient to allow the inference of L . Here we use the term somewhat loosely to define a set of examples which induces a hypothesis space containing a given logic program P . If we can show that for any arbitrary propositional Horn clause program P we can generate a characteristic sample set, it follows that there is a sample set from which any P can be induced. This in turn would show that given a large enough set of examples, which in the limit must contain a characteristic sample, the Duce operators are sufficient to learn any propositional Horn clause program.

Definition 8 Given a propositional Horn clause logic program P we say that E is a *characteristic sample* of P for algorithm $Duce_{(Abs, Intra)}$ if and only if E is a legitimate example set of P and $P \in Duce_{(Abs, Intra)}(E)$.

Before showing how to construct a finite characteristic sample for any logic program we will introduce the auxiliary notion of an isolated reference.

Definition 9 The clause $(h \leftarrow \alpha p) \in P$ is said to reference predicate symbol p .

Definition 10 The clause $C \in P$ contains an isolated reference to predicate symbol p if and only C is the only clause within P which references p .

Remark 1 If Abs^{-1} is applied to $P \supseteq \{C_1, C_2\}$ to produce $P' = P - \{C_2\} \cup \{C_1.C_2\}$ then the operator Abs^{-1} reduces the number of clauses which reference predicate symbol $p \in vocab(P)$ by one.

Remark 2 If $Intra^{-1}$ is applied to $P \supseteq (A \cup BB)$ to produce $P' = P - (\{A\} \cup BB) \cup CC$, where $A = (h \leftarrow \alpha p)$ contains an isolated reference to p , $BB = \{(p \leftarrow \beta_1), \dots, (p \leftarrow \beta_n)\}$ is the set of all clauses containing the predicate symbol p in their heads and $CC = \{(h \leftarrow \alpha\beta_1), \dots, (h \leftarrow \alpha\beta_n)\}$ then the program P' does not contain the predicate symbol p .

The following algorithm $Char_{(Abs, Intra)}$ can be used to generate a characteristic sample of a given propositional Horn clause logic program P .

```

algorithm  $Char_{(Abs, Intra)}(P)$ 
  let  $i = 0, P_0 = P$ 
  until  $P_i$  is a legitimate example set of  $P$  do
    if  $\exists A \in P_i$  such that  $A$  contains an isolated reference to  $p$ 
       $P_{i+1}$  is the result of applying  $Intra^{-1}$  to remove  $p$  in  $P_i$ 
    else
       $P_{i+1}$  is the result of applying  $Abs^{-1}$  to remove reference  $A$  in  $P_i$ 
    let  $i = i + 1$ 
  done
  let  $f = i$ 
   $E$  is  $P_f$ 
  return( $E$ )
end  $Char$ 

```

Now we must show that this algorithm will generate a characteristic sample for any propositional Horn clause logic program.

Theorem 1 $Char_{(Abs, Intra)}(P)$ returns a characteristic sample for any propositional Horn clause logic program P .

Proof. Let $E = Char_{(Abs, Intra)}(P)$. According to definition 8 E is a characteristic sample of P for $Duce_{(Abs, Intra)}$ if and only if E is legitimate and $P \in Duce_{(Abs, Intra)}(E)$. Let us assume that E is not a characteristic sample of P .

We will first look at the case in which the until loop in $Char_{(Abs, Intra)}$ terminates. According to the loop termination condition, P_f must be a legitimate example set of P . Since each step i in the derivation $P \rightarrow \dots \rightarrow P_f$ was carried out by either Abs^{-1} or $Intra^{-1}$ it follows that the sequence of transformations $(E = P_f) \rightarrow \dots \rightarrow P$ is an inverse derivation of P from E . It follows from definition 7 that $P \in Duce_{(Abs, Intra)}(E)$, and thus E is a characteristic sample of P . We must therefore assume that the until loop does not terminate.

Let p be some predicate symbol in $vocab(P) - prim(P)$. By definition there must be clauses which reference p in P . These references will be reduced one by one by the else statement (Remark 1), with the last reference being removed by the if statement, together

with all remaining occurrences of p in P (Remark 2). Referenced predicate symbols will be removed one by one until only unreferenced predicate symbols remain for some P_j . From repeated application of Lemma 1, P subsumes P_j . Moreover the vocabulary of P_j will have been successively reduced from that of P by applications of $Intra^{-1}$ (Remark 2). Thus by definition P_j is legitimate and the until loop will terminate with $f = j$. This contradicts the assumption and completes the proof. \in

We now investigate the size of the characteristic sample set for a given propositional Horn clause logic program.

Theorem 2 *Let $E = Char_{(Abs, Intra)}(P)$ and Ps be the set of referenced predicate symbols in P . The size of the characteristic sample set $|E| = |P| - |Ps|$.*

Proof. From definition 2 Abs^{-1} applies the transformation $P' = P - C_2 \cup C$, and therefore $|P'| = |P|$. From definition 4, $Intra^{-1}$ applies the transformation $P' = (BB \cup \{A\}) \cup CC$, where $|BB| = |CC|$. It follows that for $Intra^{-1}$, $|P'| = |P| - 1$. In the proof of Lemma 1 we have shown that referenced predicate symbols are removed one by one using $Intra^{-1}$. All other transformations employ Abs^{-1} . Since there must therefore be $|Ps|$ applications of $Intra^{-1}$ it follows that $|E| = |P| - |Ps|$. \in

Thus not only have we shown that the operators Abs and $Intra$ are sufficient to learn any arbitrary propositional program but also, surprisingly, less examples are needed to induce a propositional program than there are clauses in that program. This is counter-intuitive to the normal belief in inductive knowledge engineering, in which we expect to use a large number of examples to induce a small number of rules.

3.3 Search: Duce macro-operators.

Duce presently searches through the set of conjunctions of predicate symbols to find which operator to apply next. For this reason Duce can be myopic in its choice of new terms. The discrepancy in complexity between Duce's solution and human solutions (see section 2.1) seems to indicate that this problem can be quite severe. However, by considering the characteristic set generating algorithm of section 3.2 as being an inverted strategy for propositional program construction, we have discovered a simple and effective method of improving Duce's present search mechanism. The argument is as follows. The algorithm $Char_{(Abs, Intra)}$ removes intermediate concepts (predicate symbols) one at a time. For every predicate symbol p removed, $Char_{(Abs, Intra)}$ applies Abs^{-1} repeatedly to remove all but the last reference to p . p is finally removed from the vocabulary by application of $Intra^{-1}$. In reverse this strategy becomes

1. Introduce p using $Intra$
2. Apply Abs to all clauses with head p .

This can be view as a form of macro-operator. Attempts are presently being made to implement this and other macro-operators within Duce. One severe impediment to the approach has been that whereas the symbol reduction effect of the old Duce operators can be efficiently and accurately computed using the characteristic equations of [4], no efficient method of computing the exact effect of macro-operators has been found outside application and measurement. It is estimated that application and measurement would slow the execution of the Duce algorithm by a factor of 1000 on applications the size of the KPa7KR experiment. However, various methods of approximating the evaluation have been tried, the most effective of which led to a 20% compaction of the KPa7KR result [4].

4 Discussion.

Although much progress has been made in applying Duce to various problem domains, the present aim of our research is to extend Duce's capabilities. For these purposes it has been necessary to work out the theory underlying the Duce approach in more detail. In so doing we have discovered that Duce is a form of inverse resolution theorem prover. Many useful insights into possible improvements and extensions of the Duce algorithm have resulted from this, some of which are described in section 3.

The two most interesting adaptations of Duce seem to lie in the directions of changing the underlying knowledge representation to first-order predicate calculus and dealing with noisy data. Although other authors have looked at related problems [8,12,10], all such attempts have dealt with learning single predicates, none with the more difficult problem of automatic vocabulary extension.

References

- [1] D. Angluin. Inference of reversible languages. *JACM*, 29:741-765, 1982.
- [2] M. E. Bain. *Specification of attributes for computer induction*. TIRM, The Turing Institute, Glasgow, 1987.
- [3] R. Michalski and J. Larson. *Selection of most Representative Training Examples and Incremental Generation of VL1 Hypotheses: the underlying Methodology and the Description of Programs ESEL and AQ11*. UIUCDCS-R 78-867, Computer Science Department, Univ. of Illinois at Urbana-Champaign, 1978.
- [4] S.H. Muggleton. Duce, an oracle based approach to constructive induction. In *IJCAI-87*, pages 287-292, Kaufmann, 1987.
- [5] S.H. Muggleton. *Towards constructive induction in first-order predicate calculus*. TIRM, The Turing Institute, Glasgow, 1987.
- [6] G.D. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, August 1971.
- [7] J.R. Quinlan. Discovering rules by induction from large collections of examples. In D. Michie, editor, *Introductory Readings in Expert Systems*, pages 33-46, Gordon and Breach, London, 1979.
- [8] J.R. Quinlan. Learning from noisy data. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning Volume 2*, Kaufmann, Palo Alto, CA, 1986.
- [9] J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23-41, January 1965.
- [10] C. Sammut and R.B Banerji. Learning concepts by asking questions. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach. Vol. 2*, pages 167-192, Kaufmann, Los Altos, CA, 1980.
- [11] A.D. Shapiro. *The Role of Structured Induction in Expert Systems*. PhD thesis, Edinburgh University, 1983.
- [12] E.Y. Shapiro. *Algorithmic program debugging*. PhD thesis, Yale University, 1982.

Machine acquisition of concepts from sample data¹

Donald Michie
and
Michael Bain

The Turing Institute
Glasgow, UK

Abstract

Machine learning today is directed towards improved automation of the extraction of knowledge-bearing rules from data. Three typical sources of training materials for learning machines are (1) expert-supplied data, (2) raw observational data and (3) data generated from logical specifications and from simulations. Two laboratory exercises using machine-generated data are reported and their implications reviewed with regard to wider objectives of automated discovery.

¹ to appear in Artificial Intelligence and Intelligent Tutoring Systems:
1989 Spring Symposium at the University of Maine

Recently methods have been found for enabling machine-executable rules to be automatically induced from examples of expert decision-taking. This can be done under constraints which ensure that machine-learned rules are transparent to the user. Methods of this type are collectively known as machine learning, broadly defined in Figure 1.

system uses sample data (training set)

to generate an updated basis

for improved performance

on subsequent data

Figure 1. Broad definition of machine learning.

The Figure's criterion would admit to the machine learning category a variety of optimisation techniques such as adaptive control, real-time data management, signal tracking and filtering, statistical pattern recognition and the like. But the scope of machine learning is in practice more narrowly drawn, and currently tends to be confined to the topic list of Figure 2. In the case of the last item, AI-type systems, the above-mentioned transparency requirement is central.

-
- neural nets
 - genetic algorithms
 - AI-type systems
-

Figure 2. Current scope of machine learning.

The knowledge approach

In explaining what is meant by "AI-type" we start from a dictum of John McCarthy's, namely that until one has figured out a way to tell a machine things in the given domain of discourse it is not reasonable to ask it to learn them for itself. In other words a common representation has to be found. Now we turn the dictum around, and say that until one has developed a way for the machine to tell us what it has learned, its learning is not going to seem very interesting. This leads to a "strong" criterion of machine learning, as shown in Figure 3 and corresponds to what is sometimes called the knowledge approach. The idea is to go beyond mere abstraction from raw data of the "updated basis", to use the terminology of Figure 1, and to demand that the machine expresses its abstractions in terms intelligible to the human practitioner. Mountainous compilations of selected facts will not do as a representation of what has been learned, nor will compressed formulations meaningless to human intuition. The mechanised learner must generalise over its discoveries in ways which yield clear and simple rules, potential additions to its partner's mental furniture. This is easier said than done, especially when learning must be done from raw data without the help even of expert-supplied definitions of relevant low-level features.

system satisfies weak criterion
and also
can communicate its internal updates
in explicit symbolic form

Figure 3. Strong criterion of machine learning

First we give an illustration taken from a seemingly simple recognition task of just how much the addition of such features can smooth the path of propositional-level learning systems. Then we illustrate the additional power obtainable from use of a predicate logic formalism. Later we consider tasks which extend beyond recognition into difficult areas of dynamical control, where it is not feasible to supply the machine with anything beyond the means of sensing the flux of incoming data and extracting what it can. Such tasks include the automatic control of systems for which robustness and powers of adaptive self-recovery are desired to buffer against unpredicted departures from normal operating conditions.

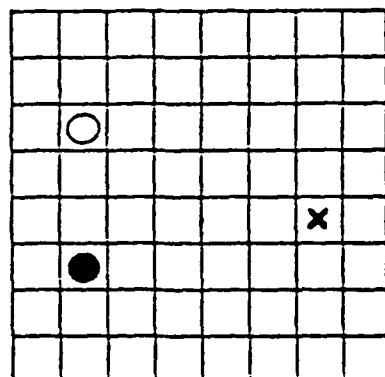
First task: building a recogniser from raw data

In the first experiment to be discussed, the learning system has to find a rule to explain a sample of a few hundred observations, each taking the form of a string of 6 integers with either a "yes" or a "no" label affixed. The rule must satisfactorily account for the observed labelling, i.e. what property of these strings determines the labelling? Having conjectured such a theory, the system tests it on new data sampled from the same source.

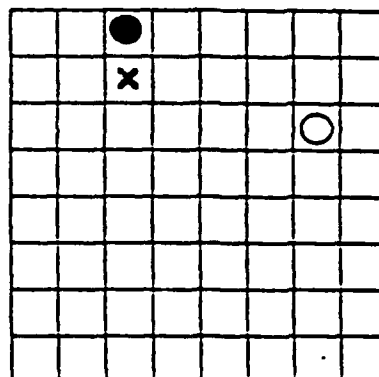
One could say that it plays the role of an empirical scientist, extracting laws from observations with the object of improved prediction of future observations. To get a rough idea from a human perspective of the learnability of the task, four intelligent teenage office workers were presented with it, both in the representation of Figure 4 and also in a more perspicuous visual format shown in Figure 5. Their responses are analysed and discussed elsewhere (Bain, Hayes-Michie, Michie and Muggleton, 1989). Sufficient to say that when faced with raw data in a form similar to that of Figure 4 they were unable to extract any meaningful patterns after attempts lasting several hours. The visual representation proved much more tractable, but this will not be further discussed here.

2	6	7	4	2	3	no
7	6	3	7	3	8	yes
3	6	1	2	6	3	no
4	8	3	2	4	7	yes
8	8	5	4	8	2	no

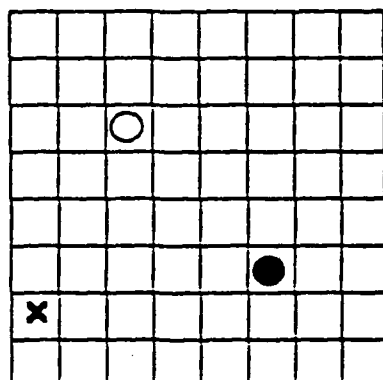
Figure 4. Some legal and illegal positions in the KRK endgame, in one of the encodings used for presentation to human subjects.



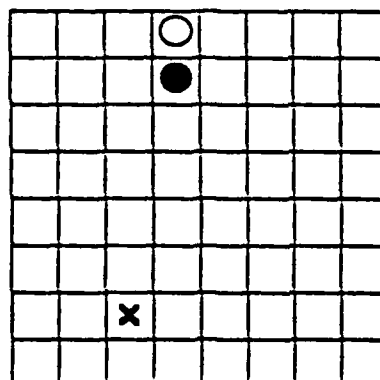
NO



YES



NO



YES

Figure 5. The first four positions in the KRK endgame which appear in Figure 4, as presented visually to human subjects.

The meaning of these records and of their classification into "yes" and "no" is shown in Figure 6. The strings of numbers are co-ordinate pairs locating on a chessboard the three pieces white king, white rook and black king. On the assumption that White has the move, "yes" designates an illegal position and "no" a legal position.

2 6 7 4 2 3 no
means WK at b6, WR at g4 and BK at b3 is legal.

7 6 3 7 3 8 yes
means WK at g6, WR at c7 and BK at c8 is illegal.

Figure 6. Attributes and classes for two KRK illegality training examples with their meanings given beneath. WK = "White King", WR = "White Rook", BK = "Black King" and b6 = "file b, rank 6".

A perfect score would be achieved by an empirical learner that generated a succinct logic formula corresponding exactly to the appropriate extract from the laws of chess. An "ideal" rule might look something like the following, expressed in chess language rather than the "rows, columns, circles and crosses" language expected from a human learner if the task were presented in Figure 5.

1. if the white rook and black king
2. either occupy the same file
3. and the white king is not directly between
4. or occupy the same rank
5. and the white king is not directly between
6. then the position is illegal;
7. if the two kings
8. either are vertically adjacent to each other
9. or are horizontally adjacent to each other
10. or are diagonally adjacent to each other
11. then the position is illegal;
12. if two pieces are on the same square
13. then the position is illegal;
14. otherwise the position is legal.

Note that lines 12 and 13 could have been transcribed directly from the laws of chess, but that the other two rules represent logically deducible specialisations from the laws. It is also interesting to note that the partial rule expressed in the lines numbered 1, 2, 4, 6 and 14 is sufficient by itself to raise the frequency of correct prediction to around 90% from the 66% baseline obtained by always guessing "legal".

The four human subjects had been selected as lacking previous contact or knowledge of the game of chess, so that in this respect they enjoyed no special advantage over the induction systems employed in the experiment. The first learning system to be tested was a state-of-the-art induction package, XpertRule (Attar Software/ITL, 1987). It uses J.R. Quinlan's (1979, 1986) ID3 algorithm to recursively partition a training set of multi-attribute records into sub-sets and sub-sub-sets, thus forming a classification tree. It incorporates a version of Quinlan's "pessimistic pruning" for removing branches far from the root where the tree's efficacy tends to be nullified by the data's intrinsic incompleteness or "noise".

Induced trees, the system's "solutions" to the problem, are then tested against new cases sampled from the same population. Duplicate trials were made, on each occasion forming a random partitioning of a file of 999 instances into a training set of 698 cases and a test set of 301. The 698 training examples constitute a very small sample of the space of positions. The latter has a size of 64^3 , or about a quarter of a million positions in total, of which about one third are illegal.

The six "primitive" measurements were augmented by fifteen mechanically derived quantities to serve as additional attributes. These were manufactured by forming all possible pairwise differences among the first six, and constitute a form of "background knowledge". Inducing with two different random partitions of the 999 instances into 698 training cases and 301 test cases gave trees of 99% and 97% measured accuracy, with 49 nodes and 41 nodes respectively. As can be seen from these performance results substantial learning occurred. The first of the above-mentioned trees is shown in Figure 7. Although the tree is ramified and rather formless, the fact that it correctly classifies 99% of the 301 test cases shows that the tree must capture most of the content of the underlying law.

A striking feature of Figure 7 is that the only attributes selected for incorporation in the tree are the three pairwise differences among attributes 1, 3 and 5 and the three pairwise differences among attributes 2, 4 and 6. Exactly the same feature characterised the other of the two induced trees (a third, generated under slightly different parameter settings was also like this, but with a few scattered exceptions). This second tree, shown in Figure 8, is slightly more compact, with a marginally lower accuracy as measured on the test set.

```

A3-A5
< -0.5 : A4-A6
  < -0.5 : A1-A5
    < -1.5 : no
    >= -1.5 : A2-A6
      < -1.5 : no
      >= -1.5 : A1-A5
        < 1.5 : A2-A6
          < 1.5 : yes
          >= 1.5 : no
        >= 1.5 : no
      >= -0.5 : A4-A6
        < 0.5 : yes
        >= 0.5 : A2-A6
          < 1.5 : A1-A5
            < -1.5 : no
            >= -1.5 : A1-A5
              < 1.5 : A2-A6
                < -1.5 : no
                >= -1.5 : yes
              >= 1.5 : no
            >= 1.5 : no
          >= -0.5 : A3-A5
            < 0.5 : yes
            >= 0.5 : A4-A6
              < 0.5 : A4-A6
                < -0.5 : A2-A6
                  < -1.5 : no
                  >= -1.5 : A2-A6
                    < 1.5 : A1-A3
                      < 0.5 : A1-A5
                        < -1.5 : no
                        >= -1.5 : A1-A5
                          < 1.5 : yes
                          >= 1.5 : A2-A4
                            < 0.5 : yes
                            >= 0.5 : no
                        >= 0.5 : no
                      > 1.5 : no
                    >= -0.5 : yes
                  >= 0.5 : A1-A5
                    < 1.5 : A2-A6
                      < 1.5 : A1-A3
                        < -4.5 : no
                        >= -4.5 : A2-A6
                          < -1.5 : no
                          >= -1.5 : yes
                        >= 1.5 : no
                      >= 1.5 : no
                    >= 1.5 : no
                  >= 1.5 : no
                >= 1.5 : no
              >= 1.5 : no
            >= 1.5 : no
          >= 1.5 : no
        >= 1.5 : no
      >= 1.5 : no
    >= 1.5 : no
  >= 1.5 : no

```

Performance : 98.7% correct on test set of 301 examples.

Figure 7. 49-node XpertRule decision-tree induced from 698 training examples described in terms of 6 primitives plus 15 pairwise difference attributes. Only pairwise difference attributes appeared as selectors in the tree, and of the 15 possible pairings just six were picked out by the algorithm.


```

A4-A6
  < -0.5 : A3-A5
    < -0.5 : no
    >= -0.5 : A3-A5
      < 0.5 : yes
      >= 0.5 : A2-A6
        < -1.5 : no
        >= -1.5 : A2-A6
          < 1.5 : A1-A5
            < 1.5 : A1-A5
              < -1.5 : no
              >= -1.5 : yes
                >= 1.5 : A2-A4
                  < 0.5 : yes
                  >= 0.5 : no
            >= : no
          >= -0.5 : A4-A6
            < 0.5 : yes
            >= 0.5 : A3-A5
              < 3.5 : A3-A5
                < -0.5 : A2-A6
                  < 1.5 : A1-A5
                    < -1.5 : no
                    >= -1.5 : A2-A6
                      < -1.5 : no
                      >= -1.5 : A1-A5
                        < 1.5 : yes
                        >= 1.5 : no
                    >= 1.5 : no
                  >= -0.5 : A3-A5
                    < 0.5 : yes
                    >= 0.5 : A1-A3
                      < 0.5 : A1-A5
                        < -1.5 : no
                        >= -1.5 : A2-A6
                          < 1.5 : A2-A6
                            < -1.5 : no
                            >= -1.5 : yes
                              >= 1.5 : no
                          >= 0.5 : no
                        >= 3.5 : no
                      >= 0.5 : no
                    >= 3.5 : no
                  >= 3.5 : no
                >= 3.5 : no
              >= 3.5 : no
            >= 3.5 : no
          >= 3.5 : no
        >= 3.5 : no
      >= 3.5 : no
    >= 3.5 : no
  >= 3.5 : no

```

Performance : 96.7% correct on test set of 301 examples.

Figure 8. 41-node XpertRule decision-tree induced from 699 training examples described in terms of 21 attributes (6 primitives, 15 differences).

Translations of such decision trees into production-rule form can be obtained automatically with great efficiency and freedom from redundancy by Quinlan's (1987) C4 algorithm. XpertRule has a similar but much weaker feature of the same general kind.. The first of eight production rules

obtained from the tree of Figure 8 was also the simplest, namely:

```
(1) IF A4-A6 >= - 0.5  
    AND A4-A6 < 0.5  
    THEN Outcome is yes
```

Remembering that even indices encode the file number and odd indices encode rank number on our chess-board, and that the piece-co-ordinate pairs are listed in the order white king (WK), white rook (WR), black king (BK), this rule simply says: "if WR and BK are on the same rank then the position is illegal". But notice how strained is the expression of even this predicate, as a result of the poverty of decision trees as a descriptive language. Apart from failure succinctly to handle equality, as in this case, decision trees run into other troubles through lacking variables. The reader may try the exercise of expressing in this restricted language (without extending the vocabulary of 21 "given" attributes) the needed refinement of the above-displayed rule, to the effect that WK must not be directly between WR and BK. He will find that no acceptably short refinement is possible. Further the foregoing inductions received considerable help from the provision of manufactured attributes, of which the critical six differences represent in effect powerful background knowledge.

The limitations of expressivity became even more obvious when we attempted XpertRule-mediated induction on files of instances described solely in terms of the original six primitives. When XpertRule was required to learn solely from the raw data, using the six primitive attributes only, typical results were as in Figure 9. Some learning has possibly occurred in the sense of Figure 1. But any learning is so weak as to be virtually non-existent.

CIGOL's output is an executable recognition rule expressed in the Prolog logic programming language. Figure 10 shows the best-performing (and best-looking) rule obtained from five CIGOL learning runs, each on a training set of 40 positive and 60 negative examples. The size of these data sets is one tenth of that needed by XpertRule when similarly confined to primitives (Figure 9), yet performance is better. The reason lies in the expressive power of the predicate calculus language and the wider repertoire of CIGOL's induction operators. Use of manufactured attributes in the XpertRule trials bars these from valid comparison with the primitives-only CIGOL results.

CIGOL-generated rule	Interpretative comments
illegal(b,2,g 6,a,3).	The first five lines contain individual instances over which CIGOL did not construct any generalisation: actually they are all cases of king-adjacency.
illegal(e,2,c,6,e,3).	
illegal(f,4,b,4,e,5).	
illegal(g,3,f,7,h,2).	
illegal(g,7,a,4,f,6).	
illegal(A,B,C,D,A,B).	WK and BK on same square;
illegal(A,B,C,D,C,E).	WR and BK on same file;
illegal(A,B,C,D,E,D).	WR and BK on same rank.

Performance : 91.4% correct on test set of 1000 examples.

Figure 10 CIGOL predicate induced from less than 40 positive training examples with no additional background knowledge. Applying negation by failure allows classification as legal all positions not known to be illegal.

There is clearly far to go before we can expect complete machine decipherment of laws hidden in the raw data of even smallish problems such as this. CIGOL's main weakness in this experiment is that it lacks a curb for its tendency to overgeneralise, and this in turn flows from the fact that its powers of generalisation are one-sided. It can only generalise over

positive instances. Although it can remember and use counterexamples as individual cases, it cannot construct a sub-rule to summarise these cases. Possible ways of remedying this are under consideration. CIGOL could then endow itself with its own critic, just as in experimental science the investigator seeks refutations for each step before committing to it.

Second task: building adaptive controllers for unstable systems

The second laboratory exercise was a dynamical control task, namely to balance (in simulation) two poles, one on top of the other, without violating certain bounding constraints. Here the system is called on to respond more in the spirit of the technologist than the scientist. Attainment of control by the learning system is regarded as primary, and acquisition of understanding in the causal sense as secondary.

We have been studying the control of a motor-driven cart on a bounded track in such a way as to keep a rigid pole (in one variant two poles end-to-end) in more or less vertical balance without running off either end of the track. The one-pole variant was the subject of experiments in the 1960's on trial-and-error learning by Michie and Chambers (1968; also Chambers and Michie, 1969). Their "BOXES" algorithm formed a foundation for the new work. By use of specific tasks such as pole balancing, alternative learning algorithms can be evaluated. For example Sammut (1988) tested and discussed procedures developed by American authors according to the "neural net" paradigm. Others belong to a category sometimes called "genetic". The name comes from an analogy with the evolution of biological species through processes of mutation and natural selection. BOXES can be seen as a distant relative of the genetic category.

We envisage a class of controllers with functionality

$$f : \text{state vectors} \longrightarrow \text{actions}$$

in which the controlling system can expect its every action to be followed

after a fixed time interval by a new input vector, to which it must again respond instantaneously with an action, or terminate in the case of FAIL.

For the single-pole task, state vectors have components <cart-position, cart-velocity, pole-angle, angular velocity>. The set of actions is {left, right}. The state components cart-position (x) and pole-angle (θ) are represented either by a real number or by the symbol FAIL signifying an out-of-range value. For cart-velocity (\dot{x}) and angular velocity ($\dot{\theta}$) out-of-range failure values are not defined. On receipt of the first vector containing FAIL the control computation gives a failure signal and terminates.

The BOXES algorithm incorporates no prior knowledge, explicit or implicit, about the nature of the system to be controlled, beyond what is conveyed by the above functionality statement together with a principle for ordering controllers with respect to merit. A controller which confers a longer expected lifetime before failure has more merit than a controller which confers a shorter expectation. Expected lifetimes are measured as means (weighted means as will later be seen) over samples of trial runs, of the number of simulated seconds between start and termination. Each test run is started from a point selected at random from a defined central region of the state-space, known as the "test range". For learning trials as opposed to test runs, random starts are taken from a larger region known as the "training range".

BOXES interprets each state vector as defining a point in an n -dimensional space, the dimensions corresponding to the sensed variables. Thus $n = 4$ for the 1-pole problem as described above. For the 2-pole problem we write θ_1 and θ_2 for the lower and upper pole angles respectively. Here $n = 6$ (x , \dot{x} , θ_1 , $\dot{\theta}_1$, θ_2 , $\dot{\theta}_2$). A controller is envisaged as a look-up table which for each point in the space contains a logical value specifying the action to be taken. In pole-balancing the action is to go left or to go right.

With point-by-point representation and infinite precision such a table would require infinite store. The BOXES algorithm approximates by partitioning the state-space so that points within a region are mapped to the same decision. For example Sammut's (1988) single-pole learning trials were conducted with partitions set by dividing the four state variables into, respectively, 3, 3, 6 and 3 intervals, as follows:

cart position (x)	$\pm 0.8, \pm 2.4$ m
cart velocity (\dot{x})	$\pm 0.5, \pm \infty$ m/sec
pole angle (θ)	$0, \pm 0.017, \pm 0.100, \pm 0.200$ radians
angular velocity ($\dot{\theta}$)	$\pm 0.87, \pm \infty$ radians/sec

This creates 162 "boxes" (i.e. $3 \times 3 \times 6 \times 3$) which fill the problem space. With each box a decision, "left" or "right", is associated. Initially, i.e. before a learning trial, these decisions are set at random (in another variant all were set to "left"). In a run-time test, after each time-step (sense-act cycle) the system arrives at a new point in n-dimensional space. The decision setting of the box within which that point is located determines the next action. For the two-pole problem, exactly the same principles apply to run-time tests, except that the dimensionality increases from 4 to 6, as earlier noted. Corresponding thresholds were set as below, yielding a total of 486 "boxes" (i.e. $2 \times 3 \times 3 \times 3 \times 3 \times 3$). In addition faster decision-cycles were employed in the experiments, with 200 Hz finally adopted as standard.

cart position (x)	$0, \pm 2.4 \text{ m}$
cart velocity (x-dot)	$\pm 0.1, \pm \infty$
lower pole angle (theta-1)	$\pm 0.017, \pm 0.200$
lower angular velocity (theta1-dot)	$\pm 0.05, \pm \infty$
upper pole angle (theta-2)	$\pm 0.00009, \pm 0.20000$
upper angular velocity (theta2-dot)	$+ 0.007, \pm \infty$

Now consider the phase preceding run-time testing, in which the system, starting from the random settings which define the initial, random, strategy, makes incremental modifications of decision settings in the light of accumulated experience so as to build an improved strategy. This is the "learning" phase. The idea behind the learning algorithm is to treat the local region of state-space defined by each box as a small world of its own. We will call such a small world, or set of states, a "situation". Each box is equipped with an independent process for maintaining statistical summaries of the average further lifetime of the system on those occasions when in this same situation it executes a "left", as against the average lifetime on those occasions when it executes a "right". At any given stage of the learning process, the setting of each box is decided by evaluating the respective mean lifetime scores of the two candidate actions. In this context, "lifetime" is counted not from the start of the run but from the execution of the "left" or "right" decision associated with the given box. The scoring function for comparing the merits of "left" and "right" takes into account that the effects of a decision are qualified by the current decision-settings of other boxes, which are themselves engaged in similar learning. Therefore average lifetimes are computed after weighting past observations for recency on the principle that in an evolving world the present resembles the recent past more closely than it resembles the remoter past. A further point is that the merit of an action in a learning context must not be assessed purely on the

basis of expected outcomes. An action has in addition an information-provoking aspect. In scoring alternative candidate actions, the decision function takes this into account.

At the end of a learning run the final decision-settings of the complete set of boxes defines a strategy, acquired from trial-and-error experience, in the form of an array of situation-action rules. Details are given by Michie and Chambers in the reference cited earlier. In the new work, where a second pole is balanced on top of the first, we combined, cell by cell, independently learned decision-arrays to form a master array.

Learning trials (double pole)

In the standard two-pole system both poles were of equal length, and parameters were as in Figure 11. In another variant, the lower pole was decreased by 2% in both length and mass, keeping density constant. This variation had no perceptible effect on the nature of the task, either from the standpoint of learning or of run-time performance. Its main use, in the event, was as a convenient way of introducing quasi-duplicate variants into the experimental design.

Force of motor 1 Newton	Max velocity 0.5 m/sec
Length of track 4.8m	Max permitted angle+ 0.2 radians
Length of poles 1.00 m for upper, 1.00 m or 0.98 m for lower	Sampling rate 200 Hz
Mass of poles 0.10 kg for upper, 0.100 kg or 0.098 kg for lower	Sense-to-act lag nil
Mass of cart 1.0 kg	Act-to-sense lag 5 msec

Figure 11. Constants of the simulated two-pole apparatus. Friction, slop, elasticity etc. not incorporated.

A typical learning trial consists of a series of "attempts" by the BOXES system, each initiated from a point randomly selected within the training range of starting values and terminated by the occurrence of a FAIL value or by the execution of 20,000 successive actions without a FAIL. The latter occurrence marks the end of the given trial, and a record is made of the number of attempts, and of the decision-settings of the boxes at the end of the trial. Obviously the smaller the number of trials, the faster the learning to criterion is judged to be. The bounds of the training range are defined below:

x from -2.4 to +2.4 m.

\dot{x} from -0.1 to +0.1 m/sec

θ from -0.2 to +0.2 radians (both lower and upper)

$\dot{\theta}$ from -1.2 to +1.2 radians/sec (both lower and upper).

Thirty two replicate trials were carried out, from each of which an array of 486 final decision-settings is harvested. A master decision-array was then compiled. Into each of its 486 cells was entered the majority-vote value extracted from the corresponding cells of the 32 arrays harvested from the learning trials. This BOXES-generated strategy was then tested.

Run-time performance tests

For testing the performance of a master decision-array, the decision-settings of the BOXES program were frozen to those of the master. Twenty successive attempts were then made from starting values selected at random from the "test range" shown below, each attempt being terminated either on receipt of a FAIL, in which case the lifetime until failure is recorded, or on attainment of a lifetime of 100 simulated seconds. Test-range bounds were:

x from -1.0 to +1.0 m

x-dot set to 0.0 m/sec

theta-1 from -0.001 to +0.001 radians (lower)

theta-2 from -0.00001 to +0.00001 radians (upper)

theta1-dot and theta2-dot both set to 0.0 radians/sec

Recapitulating the complete train-and-test cycle, the procedure was as follows:

1. Run BOXES to the 100-second criterion from 32 starts sampled at random from the training range, each time storing the final settings.
2. From the majority vote for each box, construct a single "master array".
3. Test the master array to the 100-sec limit from each of 20 starts sampled from the test range.

Results obtained with the above procedure are set out in Figure 12.

pole-length ratios	100:100	98:100	overall means
--------------------	---------	--------	---------------

BOXES: no. of learning attempts

before first 100-sec success (means of 32 trials)

361, 290, 289, 391; 265, 306, 320, 259; 310.1

BOXES-derived rules: run-time tests;

mean lifetimes 41.7, 78.5, 34.8, 81.3; 19.2, 74.1, 99.0, 17.6; 55.8

100-sec successes/20 0/20,14/20,0/20,11/20; 0/20, 12/20,17/20,0/20; 54/160

Figure 12. Induction of two-pole control rules. Sampling rate = 200 Hz.

Results of four replicates for each of the two variants "100:100" and "98:100".

This "trial and error" machine learning contrasts with derivation of rules by hand from first principles. The latter approach was exemplified by Makarovic's (1989) study in which he developed a two-pole balancer in top-down style from an analysis of the equations of motion. He found the process almost prohibitively laborious, and was obliged to adopt numerous approximations and simplifying assumptions. In the result, the Makarovic rule (Figure 13) is elegant, but in common with the BOXES-derived rule it cannot be certified correct for all possible states. Both methods, the data-oriented and the theory-oriented, were, however, shown to be capable of generating controllers of the two-pole system which would maintain control for more than 100 seconds of simulated time. Naturally the ultimate goal of both investigations is a methodology which can deliver complete certifiability of rules.

```

if  $\dot{\theta}_1$  = big positive then Left
if  $\dot{\theta}_1$  = big negative then Right
if  $\dot{\theta}_1$  = small
  then if  $\theta_1$  = big positive then Left
    if  $\theta_1$  = big negative then Right
    if  $\theta_1$  = small
      then if  $\dot{\theta}_2$  = big positive then Right
        if  $\dot{\theta}_2$  = big negative then Left
        if  $\dot{\theta}_2$  = small
          then if  $\theta_2$  = big positive then Right
            if  $\theta_2$  = big negative then Left
            if  $\theta_2$  = small
              then if  $\dot{x}$  = big positive then Right
                if  $\dot{x}$  = big negative then Left
                if  $\dot{x}$  = small
                  then if  $x$  = positive then Right
                    if  $x$  = negative then Left

```

Figure 13. The Makarovic rule, hand-inferred from the equations of motion of the double pole system; "big" and "small" are defined by numerical thresholds, not shown here.

Two-pole balancing: the null action

As already stated, for the two-output version of the pole-and-cart problem ("left" and "right"), the BOXES system proved capable of creating structured decision models with reasonable run-time behaviour. We were, however, more interested in the realistic case where the available actions include the null action, — i.e. the options are "left", "right" and "do-

nothing". In industrial control and in aerospace the conservation of fuel expended in control actions is often a paramount concern. So we were anxious to establish that our learning system could handle the combinatorial growth of the decision space resulting from augmenting the action set.

After adding a null action to the BOXES repertoire we conducted learning trials and run-time tests as before, except that all decision settings were initialised to "null". Experiments were conducted with both 100 Hz and 200 Hz variants. Superiorities of the 200 Hz setting were again observed, qualitatively and quantitatively similar to the previous finding. The results set out in Figure 14 are for 200 Hz only.

pole-length ratios	100:100	98:100	overall means
--------------------	---------	--------	---------------

BOXES: no. of learning attempts			
before first 100-sec success (means of 32 trials)			
	301, 315, 372, 373;	469, 474, 316, 323;	367.9
BOXES-derived rules: run-time tests;			
mean lifetimes	43.1, 84.9, 72.2, 55.5;	97.9, 39.3, 65.2, 26.3;	60.5
100-sec successes/20	1/20,13/20,12/20,3/20;	16/20, 1/20,5/20,0/20;	51/160

Figure 14. Induction of two-pole control rules with null action added.
Sampling rate = 200 Hz. Results of four replicates for each of the two variants "100:100" and "98:100".

The new variant shows a small increase in the learning time, from a mean of 310.1 to a mean of 367.9, while the quality of acquired skill as

judged by the run-time tests shows no noticeable difference. Since the number of decision-options has increased from 2 per box to 3 per box, with a corresponding increase in the amount of entropy to be overcome, this is satisfactory.

The "do-nothing" category suggests a figure of merit or "objective function". Controllable regions of state-space subdivide themselves into two levels of desirability: those in which the controller can afford to coast and those in which it must occupy itself with remedial action. This is where we are presently looking for radical improvement in the algorithm. If the learning system can be modified to recognise as a "mini-FAIL" each transition from a region of state-space ("box") for which the decision-setting is currently null, to one for which it is left or right, and also to recognise a transition in the opposite direction as a "mini-SUCCESS", then learning can occur from moment to moment within a single run, without requiring the occurrence of "maxi-FAIL" states (i.e. crashes) as a prerequisite for improvement. A good controller is one which spends as much time as possible coasting. A case in point is the control of orbiting space vehicles, in which thruster and torquer fuel must last the vehicle's lifetime..

Discussion

A common thread runs through the diagnosis of illegality in the king-rook-king ending, the two-pole balancing on a motor-driven cart and control studies, not reported here, using computer simulations of an orbiting space vehicle. That thread is the search for an effective technology of discovery. In each case the laws were deliberately hidden by the experimenters. The system was then asked to discover them, or rather *their heuristic equivalents*. The use of italics in the foregoing sentence is to affix a danger flag to any assumption that causal laws and their heuristic equivalents are the same. Even for relatively simple tasks they are very different. A skilled illegality-adjudicator of king-rook-king games, accustomed to deciding at a glance, could only slow himself down by invoking his knowledge of why king-adjacency is one of the disqualifying conditions, or why interposition

of the white king reverses the import of collinearity between the other two pieces. What if a task is more complex, and must be executed within some allotted time frame? Then the slow-down involved in invoking "deep" knowledge can be fatal. We close by illustrating with a parable.

A circus in California was renowned for the skill of its trick bicycle rider. Wishing to enhance his star's prowess yet further, the manager sent him to attend a graduate course on the dynamics of the bicycle. This course, to be conducted by a world-famous professor of physics, had been advertised by a nearby university. At the end of the course the star performer returned to his trade, of which he now seemed to show diminished grasp. Soon afterwards he made an error during high-wire bicycling and left the circus.

Meanwhile the professor's class, enraptured by the excellence of his course, presented him with a de luxe racing bicycle. The professor, who had never before ridden a bicycle, mounted it. The consequent shock and injuries led him to take early retirement. He and the circus performer have now combined to lobby Congress, but find difficulty in agreeing their platform. The professor wishes to ban the possession or use of bicycles by students of control theory. The performer wants textbooks of mechanics to carry a Government health warning.

The underlying issue has been analysed elsewhere (Michie, 1979, 1982). The short conclusion is that in building skilled recognisers and controllers from raw data, learning systems must proceed along the performer's bottom-up path of inducing heuristic rules from data. We fully concede the importance of projects to automate the understanding of causal laws. But for the specific purpose of automating the acquisition of skills, adequate mechanisation of trial and error learning will remain the prime prerequisite.

Acknowledgements

Work on the inductive learning of logical predicates from chess end-game data was supported in part by research contract No. 092/167 from the Alvey Directorate of the Department of Trade and Industry. The theoretical ideas, together with the design and implementation of the CIGOL system were contributed by Dr Stephen Muggleton, to whom we owe thanks for permission to quote unpublished work. We have also enjoyed valuable collaboration from our institutional partners in this contract at Bradford University and at International Computers Ltd, Bracknell.

The work interacted closely with an investigation within the Turing Institute into skill-acquisition in ultra-complex domains. The latter was made possible by Contract number DAJA45-86-0047 from the US Army Institute for the Behavioural and Social Sciences through its European Science Coordination Office at the European Research Office of the US Army, London, England. Support and facilitation has also been received from Apple Computer International, British Aerospace and the National Engineering Laboratory.

References

Attar Software/ITL (1987) *XpertRule* (version 4.0), Manchester: Attar Software Ltd, and Glasgow: Intelligent Terminals Ltd.

Chambers, R.A. and Michie, D. (1968) Boxes: an experiment in adaptive control. In *Machine Intelligence 2* (eds. E.Dale and D.Michie), Edinburgh: Edinburgh University Press.

Makarovic, A. (1989) A qualitative way of solving the pole-balancing problem. In *Machine Intelligence 12* (eds. J.E.Hayes, D.Michie and E.Tyugu), Oxford: Oxford University Press.

Michie, D. (1979) Machine models of perceptual and intellectual skills. In *Scientific Models and Man. The 1976 Herbert Spencer Lectures* (ed. H.Harris), Oxford: Oxford University Press.

Michie, D. (1982) High road and low road programs. *AI Magazine*, 3, 21-22.

Michie, D. and Chambers, R. (1969) Man-machine co-operation on a learning task. In *Computer Graphics: Techniques and Applications* (ed. R.Parslow, R.Prowse and R.Elliott-Green), London: Plenum Publishing Co.

Muggleton, S.H. (1988) A strategy for constructing new predicates in first-order logic. In *EWSL88: Proc. Third European Working Session on Learning*. (ed. D. Sleeman), London: Pitman Publishing.

Muggleton, S.H., Bain, M., Hayes-Michie, J.E., and Michie, D. (1989) An experimental comparison of learning formalisms. Submitted to *Sixth International Workshop on Machine Learning*, Cornell University, Ithaca, NY, USA.

Quinlan, J.R. (1979) Discovering rules by induction from large collections of examples. In *Expert Systems in the Microelectronic Age*. (ed. D.Michie), Edinburgh: Edinburgh University press.

Quinlan, J.R. (1986) Induction of decision trees. *Machine Learning*, 1, 81-106.

Quinlan, J.R. (1987) Generating production rules from decision trees. In *Proc. Tenth Int. Conf. on Art. Intell.*, Los Altos: Kaufmann.

A strategy for constructing new predicates in first order logic

Stephen Muggleton (steve@turing.ac.uk)
The Turing Institute
36 North Hanover St
Glasgow G1 2AD, UK *

Abstract

There is increasing interest within the Machine Learning community in systems which automatically reformulate their problem representation by defining and constructing new predicates. A previous paper discussed such a system, called CIGOL, and gave a derivation for the mechanism of inverting individual steps in first order resolution proofs. In this paper we describe an enhancement to CIGOL's learning strategy which strongly constrains the formation of new concepts and hypotheses. The new strategy is based on results from algorithmic information theory. Using these results it is possible to compute the probability that the simplifications produced by adopting new concepts or hypotheses are not based on chance regularities within the examples. This can be derived from the amount of information compression produced by replacing the examples with the hypothesised concepts. CIGOL's improved performance, based on an approximation of this strategy, is demonstrated by way of the automatic "discovery" of the concept of radiation. This example also demonstrates CIGOL's ability to ignore irrelevant background knowledge and deal with multiple interacting concepts.

*This work was supported by the British Government's Alvey Logic Database Demonstrator project and a grant from the US Army Research Institute for the Behavioural and Social Sciences through its European Research Office, London, England, Contract no. DAJA45-86-0047.

1 Introduction

A concept can only be learned if it can be represented. More than this, an appropriate representation language facilitates a simple and elegant description of a target concept. In [17] we describe a system called CIGOL which automatically develops its own representation language in order to efficiently represent target concepts. Initially CIGOL is provided with pertinent background knowledge in the form of Horn clauses in first order logic. CIGOL is then presented with a sequence of ground unit clauses, representing positive instances of the target concept. Following the presentation of each example CIGOL presents the user with a sequence of hypotheses which take one of two forms: either "Is X true?" or "What shall I call the following concept?". The first type of question involves a generalisation which could be used to derive previous examples. Note that each negative response from the user adds a negative instance to CIGOL's "training set" which otherwise would consist solely of positive instances. The second type allows the introduction of new relational predicates which enable the target concept to be represented more efficiently. Since these forms of generalisation are chained together, the introduction of a new predicate is typically followed by further generalisation and/or decomposition into related sub-concepts.

The generality of the approach used allows CIGOL to exhibit a number of facets of *Machine Learning*. Thus CIGOL can be classed with systems which carry out

1. **inductive concept formation** such as [12,21]
2. **constructive induction** such as [22,13]
3. **discovery** such as [10,9,6]
4. **generalisation of single examples using background knowledge** such as [5,15,24]

Unlike most learning systems described in the literature CIGOL uses an unrestricted form of first order Horn clause logic which allows predicate relations to take not only variables and constants as arguments but also complex terms. This allows CIGOL to learn not only simple structural concepts, but also more complex program fragments. The various hypothesis forming mechanisms employed by CIGOL are based on inverting individual steps of a resolution proof. This approach is a generalisation of the approaches used

by Sammut and Banerji [24], Muggleton [16] and Banerji [1]. Other strongly related work in progress can be found in Wirth [27] and Wrobel [28].

In [17] we provided a derivation for the inverse resolution operators employed by CIGOL but to a large degree left open the question of strategy of operator application. The result was that CIGOL as described in [17] proposed a number of irrelevant and uninteresting hypotheses based on the discovery of chance and unimportant regularities within the presented examples. In this paper we describe a formal framework for a strategy of operator application aimed at avoiding the generation and testing of uninteresting hypotheses. The strategy is based on results from algorithmic information theory[4]. The minimal bit size difference criterion unifies what are usually perceived as two different kinds of inductive gain. The gain which is produced by increasing the cover of a concept and the store-cost gain involved in simplifying the description of the concept, possibly involving decomposition into simpler sub-problems. An approximation to the new strategy has been incorporated into a new version of CIGOL. Sample results of the improved behaviour of CIGOL are included for a discovery problem from Francis Bacon's *Novum Organum*.

2 Generality and inverse resolution

Using standard notation from logic one can define the generality relation between well-formed-formulae F_1 and F_2 as follows

$$F_1 \text{ is more general than } F_2 \text{ iff } F_1 \vdash F_2$$

where $F_1 \vdash F_2$ should be read as " F_1 entails F_2 " or alternatively " F_2 is provable from F_1 ". Note that this simple definition allows us not only to compare the relative generality of atomic formulae and clauses but also the same relationship for arbitrary pairs of theories (sets of clauses). Buntine [3] describes an algorithm aimed at computing this generality relationship which he terms "generalised subsumption". For a fuller discussion of the subject of generality the reader is referred to Niblett [18]. Following Plotkin [19] we may more precisely define the setting of inductive learning using the following relationship

$$I \wedge B \wedge H \vdash E^+ \tag{1}$$

where I is background knowledge which is not pertinent to the present learning problem, B is background knowledge which is pertinent to the problem,

H is an hypothesis consisting of one or more clauses and E^+ is a set of positive examples. In addition, if E^- is a set of negated formulae representing counter-examples then we can guard against over-generalisation by ensuring that $I \wedge B \wedge E^- \wedge H$ is not *unsatisfiable*, i.e. self-inconsistent. The explicit definition of the inductive setting described by (1) allows us to both pose and answer the following questions concerning the approach to learning embodied in CIGOL. Since this setting is analogous to both scientific theory formation and automatic program construction the author will indulge in a certain amount of mixed metaphor in the following discussion.

Question 1 *How can we construct H given I , B , and E^+ ?*

Answer: All methods outside enumeration and testing of H rely on applying efficient "generalisation operations" which incrementally construct H from B and E^+ . Michalski [11] notes that these generalisation operations can be based on reversing the deductive rules of inference which allow us to derive E^+ from B and H . Michalski's INDUCE system uses the inversions of a wide variety of deductive rules of inference. However, we note that this is somewhat analogous to the position in theorem proving before the introduction of the universal rule of deductive inference known as resolution [23]. The thesis behind CIGOL is that appropriate inversions of resolution provide an efficient, sufficient and complete mechanism for the inductive setting described by (1).

Question 2 *Given that the predicate vocabulary used in E^+ and E^- might reasonably be limited to the "observation language" of experimentation and measurement, how do we develop a "theoretic language" of predicates which cannot be directly observed?*

Answer: Clearly the theory described by B and H can in principle contain predicates which, although relevant to the entailment of the observations E^+ and E^- , are not expressed in the vocabulary of E^+ and E^- . In CIGOL this "theoretic vocabulary" is introduced via the "W" operator (Intra-construction) (see [17]) and generalised and integrated into B using the "V" operator (Absorption).

Implementation details relating to answers 1 and 2 are given in [17]. However, familiarity with practical implementation details and the methods of scientific investigation would suggest that we must at least address the following additional questions.

Question 3 *How do we effectively constrain the generation of possible hypotheses H ?*

Question 4 *How can we judge our confidence in any particular H ?*

Question 5 *What is the criterion for distinguishing between the relevant background knowledge B and irrelevant background knowledge I ?*

In the following sections we discuss possible approaches to answering questions 3-5.

3 Search strategies and algorithmic information

3.1 Version spaces

In [14] Mitchell describes a general search strategy for inductive inference, known as the "Version space" approach. This method involves the maintenance of two sets, S and G . These sets represent respectively the least and most general hypotheses which are consistent with the examples so far. The idea is that as increasing numbers of examples are presented the space of plausible hypotheses defined by S and G converges in the limit to a singleton. At this point the system can be said to have recognised the concept.

Might this simple and attractive technique be adapted to the purpose of guiding the search in CIGOL? The answer is "no" for the following reasons. In Mitchell's description, hypotheses are single clauses constructed from a fixed language and containing no terms as arguments other than variables and constants. The generality relationship for Version spaces can be defined by saying that clause C is more general than clause D whenever $C \vdash D$ (see section 2). This generality relationship induces a *finite* lattice over the space of hypotheses, an essential pre-condition for the Version space approach to be effective. CIGOL has a less restricted form of hypothesis language consisting of arbitrary *sets* of Horn clauses. In this case the generality relationship for hypotheses becomes: theory T_1 is more general than theory T_2 whenever $T_1 \vdash T_2$ (section 2). This relationship induces an *infinite* lattice over first order Horn clause theories. Moreover, although top and bottom of the lattice can be defined by theories which are equivalent to the empty clause (the logical constant *false*) and the empty theory (*true*) respectively, according to Plotkin [19] there exist infinite length ascending and descending chains of generality within this lattice. Clearly this indicates that irrespective of considerations of computational efficiency, a Version space search could not be expected to converge within such a lattice. We must therefore look to some alternative model to guide and constrain the search through this more complex lattice.

3.2 Algorithmic information theory

Following the lead of Kolmogorov [8] various information theorists [4,26,2], have investigated the relationship between computation, randomness and message complexity. The basic intuition rests on the observation that although the strings

010100110111001100010110101100 and
010101010101010101010101010101

have approximately the same Shannon information content [25], the second contains a higher degree of regularity than the first. As an alternative to standard information measures Kolmogorov defined the algorithmic information of a finite string s as being equal to the bit length of the minimal Universal Turing machine program s^* which generates precisely s as output. Thus long regular strings have lower Kolmogorov information than strings of the same length which have no regularity. In addition, Kolmogorov defines a *random* string to be one which cannot be compressed by being encoded as a program for a reference Universal Turing machine.

By definition, inductive construction of first order theories involves a form of information compression. This follows from the fact that most finitely expressible first order theories entail an unbounded set of instances. Moreover, inductive inference from a finite set of examples can never be carried out with absolute confidence. However, we feel increased confidence in hypotheses that cover increasing numbers of examples. We will now attempt to formalise the notion of confidence in hypotheses directly with respect to compression of information.

3.3 CIGOL hypotheses and chance regularity

For the reader's convenience we provide a proof sketch for the following theorem and corollary from algorithmic information theory [4].

Theorem 1 *Let Σ_n be the set of all binary strings of length n , T_r be an arbitrarily chosen reference Turing machine and the k -bit-compressible strings of length n , $K_{n,k}$, be defined as $\{y : y \in \Sigma_n, x \in \Sigma_{n-k}, T_r(x) = y\}$. The set $K_{n,k}$ has at most 2^{n-k} elements.*

Proof Since Turing machines are deterministic T_r either induces a partial one-to-one or many-to-one mapping from the elements of Σ_{n-k} to the elements of $K_{n,k}$. Thus $|K_{n,k}| \leq |\Sigma_{n-k}| = 2^{n-k}$. \in

Corollary 2 *The probability of a binary string generated by tossing an unbiased coin being compressible by k bits using any Turing machine T_r as a decoding mechanism is at most 2^{-k} .*

Proof Applying theorem 1, the proportion of randomly generated strings which are compressible by k bits is at most $2^{n-k}/2^n = 2^{-k}$. \in

Note that T_r is merely used here for decoding compressed strings. In addition these results hold irrespective of the choice of T_r . As mentioned in the previous section information theory defines the absolute information of a string by making use of the special case in which T_r is a reference Universal Turing Machine. However, this is immaterial to the present discussion since the discovery of such a minimal-length encoding is an undecidable problem [4]. On the other hand, clearly the mere compressibility of a string relative to a particular decoding machine which is known to halt on all inputs is decidable.

Now consider again the inductive setting described by

$$I \wedge B \wedge H \vdash E^+$$

Within CIGOL, background knowledge is built up incrementally. Imagine that the theory P is built entirely on the basis of examples. Though some of P will be irrelevant to some examples, we can view P as being a single hypothesis which entails the examples. Thus

$$P \vdash E^+$$

Of course an inductive agent cannot know the origin of the examples E^+ . When evaluating the results of experimentation one generally makes use of a null hypothesis, \bar{H} , which is the negative of the hypothesis, H , being tested. By refuting \bar{H} one demonstrates the plausibility of H . In our setting we might take the null hypothesis to be that every bit in the encoding of the examples E^+ was produced by tossing a coin. Note that H is an hypothesis about an hypothesis. We can find an upper bound on the probability of the null hypothesis using corollary 2 by defining a reference Turing Machine T_r which, given an encoded version of P as input generates an encoded version of E^+ as output. Thus

$$T_r(I(P)) = O(E^+) \quad (2)$$

where $I(P)$ is an input tape encoding of P , $O(E^+)$ is an output tape encoding of E^+ and $I(P)$ is k bits shorter than $O(E^+)$. The machine T_r will be described in the next section. We will use X_k to denote the statement

that there exists such a k -bit compressed explanation $I(P)$ of $O(E^+)$. Now according to corollary 2

$$Pr(X_k|\overline{H}) \leq 2^{-k}$$

We will use the probability of $\overline{X_k}$ given that the null hypothesis was true as a measure of our confidence that the compression produced by accepting P is not based on the discovery of chance regularities within E^+ . This is

$$\begin{aligned} Pr(\overline{X_k}|\overline{H}) &= 1 - p(X_k|\overline{H}) \\ &\geq 1 - 2^{-k} \end{aligned} \tag{3}$$

Example 1 Let $O(E^+)$ be 110 bits long and $I(P)$ be 100 bits long. Then

$$\begin{aligned} Pr(\overline{X_k}|\overline{H}) &\geq 1 - 2^{100-110} \\ &\geq 1 - 1/1024 \\ &\geq 0.999 \end{aligned}$$

Note the following intuitively appealing features of (3) as a measure of hypothesis confidence. Firstly, $Pr(\overline{X_k}|\overline{H})$ is only well defined as a probability when k is positive, i.e. we can only have confidence in a theory which is less bulky than the facts on which it is based. Secondly, for all finite values of k $Pr(\overline{X_k}|\overline{H})$ is less than 1, i.e. no matter how many facts are covered by a theory, we can never have total confidence in it. Thirdly, an increase in k when it is already large provides only a small increase in $Pr(\overline{X_k}|\overline{H})$, i.e. there are diminishing returns in the confidence inspired in a theory involved in showing that it covers an increasingly large number of facts. All of these are standard assumptions within the philosophy of science [20].

It is now necessary to describe in more detail the reference Turing machine T_r and its input and output tape encodings I and O .

3.4 Encodings and the compression model

First an efficient Turing tape encoding M for any logical expression, S is described. The encoding M should be efficient in the sense that almost any tape encoding should correspond to a particular logical expression and vice versa. This is necessary for testing \overline{H} since we do not want to introduce the possibility of spurious compressibility due to inefficient encoding of the examples.

A set of Prolog clauses can be coded as a single logical expression using list concatenation symbols as necessary to separate clauses. One such coding might be to use a standard prefix coding ¹, such as Huffman codes, for coding each function symbol and variable, and write the expression on the tape using reverse polish notation. Reverse polish allows us to ignore the requirement for bracketing and separators.

Let $sym(S)$ represent the combined set of variables and function symbols of given arity within S and let N be the sum of the frequencies of occurrence of elements of $sym(S)$ within S . Now if we write the relative frequency of occurrence of symbol s in S as p_s then, ignoring the length of the prefix table, $M(S)$ has a length of

$$|M(S)| \approx N \sum_{s \in sym(S)} -p_s \log_2 p_s \text{ bits}$$

according to Shannon information theory. There is obviously a similarity here to the entropy function used in ID3 [21], which should not be surprising given the common basis in information theory.

Example 2 Let S be $[crow(harry), (black(X) :- crow(X))]$. Then $sym(S)$ is $\{':-/2, '.'/2, crow/1, black/1, X/0, harry/0, []/0\}$, $N = 10$ and the corresponding relative frequencies are $\langle 0.1, 0.2, 0.2, 0.1, 0.2, 0.1, 0.1 \rangle$. Thus $|M(S)| \approx 10(4 \times 0.1 \times 3.3 + 3 \times 0.2 \times 2.3) \approx 27 \text{ bits}$.

Clearly we can use M as the output tape encoding function O described in the previous section, rewriting (2) as

$$T_r(I(P)) = M(E^+)$$

Could we also use M as the input tape encoding function I ? In fact we cannot, since the logic program P does not contain sufficient information to describe which particular instances it was derived from. Thus in general there is no Turing machine which could take an encoding of an arbitrary logic program P and print out the set of instances E^+ from which P was derived. This leaves us in a predicament as to how to represent this additional information for the purposes of our model of hypothesis confidence.

The following is a possible solution. Devise a numbering scheme for all instances entailed by P . Now append the numbers corresponding to the

¹Prefix codes are variable length bit patterns used to encode the symbols in a message. Efficient coding schemes allow one to encode each symbol in close to the optimal of $-\log_2 p$ bits per symbol, where p is the relative frequency of the symbol within the message.

particular examples in E^+ onto the encoded description of P to make $I(P)$. Such a numbering scheme is called a Gödel numbering after [7]. A natural numbering scheme that suggests itself is to consecutively number the unit clauses in order of their first appearance within the levelwise expansion of the resolution universe, $R^*(P)$ [23]. The levels of $R^*(P)$ are defined as follows

$$\begin{aligned} R^0(P) &= P \\ R^n(P) &= R^{n-1}(P) \cup \{C : C_1, C_2 \in R^{n-1}(P), \\ &\quad C \text{ is the resolvent of } C_1 \text{ and } C_2\} \end{aligned}$$

$R^*(P)$ is simply the closure $R^0(P) \cup R^1(P) \dots$. We will use $unit(i, P)$ to denote the i th unit clause i in this enumeration. However, this numbering scheme is still not adequate for the purpose since $R^*(P)$ does not contain all the *ground* unit clauses entailed by P . However, it is straightforward to show that for every ground unit clause L entailed by P there is a unit clause L' in $R^*(P)$ and a substitution θ such that $L = L'\theta$. Thus our input tape encoding I can simply be as follows

$$I(P) = M(< P, u_1, \theta_1, \dots, u_n, \theta_n >)$$

where $unit(u_i, P)\theta_i$ is the i th example from E^+ , and M is used to encode the entire expression $< P, u_1, \dots >$.

We are now in a position to calculate the lower bound on hypothesis confidence expressed in inequality (3) of the previous section.

Example 3 Let $E^+ = \{crow(tom), crow(dick), crow(harry), crow(janis)\}$ and $P = \{crow(X)\}$. Now $I(P) = M(< [crow(X)], 0, (X/tom), 0, (X/dick), 0, (X/harry), 0, (X/janis) >)$. Computing encoding lengths in the fashion demonstrated in example 2 we find that $|M(E^+)| \approx 32$ bits while $|I(P)| \approx 65$ bits. Thus since the encoding for the hypothesis is longer than the encoding of all of the examples we can attribute no confidence to the hypothesis.

Example 4 Let $E^+ = a(b(c(d(e(f(g(h))))))), a(b(c(d(e(f(g(i))))))), a(b(c(d(e(f(g(j)))))))$ and $P = a(b(c(d(e(f(g(X)))))))$. In this case $|M(E^+)| = 96$ bits and $|I(P)| = 73$ bits. Substituting these values into (3) produces a confidence of at least $1 - 2^{-23} = 0.9999999$, i.e. virtual certainty that the compression produced by accepting the hypothesis is not accidental.

Examples 3 and 4 demonstrate an important point regarding the truncation operator within CIGOL. In the version of CIGOL described in [17]

hypotheses were preferred entirely on the basis of textual simplicity. As a result, the learning sessions demonstrated that CIGOL had a strong tendency to overgeneralise when applying the truncation operator. Since this operator merely replaces terms by variables, the most preferred description is bound to be the most general, such as *member(X,Y)*. However, since simplicity seemed to be a powerful heuristic for all other cases, it seemed arbitrary to use a different heuristic for this special case. As examples 3 and 4 demonstrate, the confidence bound described by (3) produces a much more satisfactory result, and clearly distinguishes between generalisations based on weak and strong similarities respectively.

In summary, the reader should note that the new strategy is based on preferring inverse resolution operators on the basis of their abilities to shorten the minimal achievable encoding of the examples.

4 Example sessions

Francis Bacon's *Novum Organum* (1620) is an early but thorough exposition of what is now known as scientific method. As an example of the method of hypothesis formation Bacon demonstrates that many of the properties of light can be inferred from a small set of known facts. In this section we demonstrate the performance of the revised CIGOL which employs the "confidence" statistic developed in previous sections of this paper. The example below is in the spirit of Bacon's exposition of the properties of light. User input is underlined, and excessive computer output with no corresponding input from the user is replaced by "...".

```
!- [-inverse].
!- show_clauses.
inverse(huge, tiny).
inverse(large, small).
inverse(small, large).
inverse(tiny, huge).
proportional(huge, huge).
proportional(large, large).
proportional(small, small).
proportional(tiny, tiny).
!- situation(dist(light, board, tiny),
illum(board, huge)).
Confidence = NIL for (situation(dist(light,
```

```

board, tiny), illum(board, A)):-inverse(A, tiny))
Confidence = NIL for (situation(dist(light,
board, A), illum(board, B)):-inverse(B, A))
...
!-

```

In the session so far the user started by loading background knowledge concerning the qualitative relations *inverse* and *proportional*. The user now describes an observed situation, *situation(dist(light, board, tiny), illum(board, huge))*, involving a board and a light source. In the situation described, the light source is a tiny distance from the board, and the illumination on the board is huge. CIGOL tries various hypotheses, including a possible inverse relationship between distance and illumination. However, with only a single example on which to base the hypothesis CIGOL has insufficient confidence to suggest the hypothesis to the user. In the version of CIGOL described in [17] CIGOL would at this point have chosen the hypothesis *(situation(dist(light, board, A), illum(board, B)):-inverse(B, A))* to present to the user since this is the simplest within the hypothesis space. Using the new confidence statistic, CIGOL acts more cautiously since this hypothesis, although simple, is no simpler than the presented example itself. We now continue the session.

```

!- situation(dist(light, board, large),
illum(board, small)).
TRUNCATION: (0.999)
Is situation(dist(light, board, A),
illum(board, B)) always true? n.
...
ABSORPTION: (0.99999)
New clauses:[(situation(dist(light, board, A),
illum(board, B)):-inverse(A, B))]
cover new facts: [situation(dist(light, board,
huge), illum(board, tiny)), situation(dist(light,
board, small), illum(board, large)), ...]
Are new clauses always true? y.
!-

```

Given the additional example, CIGOL finds a high confidence level (0.999) for applying "truncation" since the two examples share a lot of structure (see examples 3 and 4). However, the user rejects this and is instead presented,

by absorption, with the previously rejected hypothesis stating the inverse distance relation. Note that this hypothesis, though textually more complex than the truncation leads to an even higher confidence level using the new model. The reason for this is that no additional substitutions need be stored to describe the Gödel numbers of an absorption or intra-construction. Note also that the background knowledge concerning the qualitative *proportional* relationship was never proposed in any hypothesis. The reason again for this is that the new confidence statistic always rejects hypotheses unless they simplify the description. This provides something of an answer to question 5 in section 2 of this paper. It should be noted that the limitations of efficient search using a best first algorithm can lead to the normal problems to do with local minima.

The inverse distance hypothesis is accepted by the user. Interestingly, Bacon proposed this qualitative relation for light from simple observations more than 60 years before Newton's quantitative inverse square distance law for gravitation.

Next the user provides examples concerning the illumination on an opaque globe, resulting in the additional inverse distance law *situation(dist(light,globe,A),heat(globe,B)) :- inverse(A,B)*. CIGOL then combines these two laws by constructing a new predicate as follows.

```
INTRA-CONSTRUCTION (0.99997)
  situation(dist(light,A,B),heat(A,C)):-
    p559(A),inverse(B,C).
  p559(board).
  p559(globe).
  What shall I call p559? opaque.
!-
```

Thus CIGOL discovers a set of objects which reflect, and the user names the new concept "opaque".

The user next goes through the same process with respect to the heat properties of the light source for the given objects, eventually producing the similar rule *situation(dist(light,A,B),heat(A,C)):- opaque(A), inverse(B,C)*, i.e. opaque objects get hot when close to the light source, and are cooler when the light source is moved away. CIGOL now combines the two analogous rules together to construct a new predicate again as follows.

```
INTRA-CONSTRUCTION (0.99999)
  situation(dist(light,A,B),C):-
```

```

    p773(A,C,D),opaque(A),inverse(B,D).
    p773(A,heat(A,B),B).
    p773(A,illum(A,B),B).
    What shall I call p773? radiation.
    !- show_clauses
    inverse(huge, tiny).
    inverse(large, small).
    inverse(small, large).
    inverse(tiny, huge).
    proportional(huge, huge).
    proportional(large, large).
    proportional(small, small).
    proportional(tiny, tiny).
    opaque(board).
    opaque(globe).
    situation(dist(light,A,B),C):-
        radiation(A,C,D),opaque(A),inverse(B,D).
    radiation(A,heat(A,B),B).
    radiation(A,illum(A,B),B).

```

CIGOL has managed to combine the concepts of light and heat to produce a new 3-place relation which the user calls *radiation*. The three arguments of radiation correspond respectively to the *transmitter*, *radiation-type* and *receiver*. Note that this predicate construction is rather like a second order *analogy*, even though CIGOL works only in first order logic. The reason CIGOL managed to carry out a pseudo-second-order analogy is because the properties *heat* and *illumination* were described within the examples using function symbols rather than predicate symbols.

At the send of the session the user types *show_clauses* to reveal the entire set of clauses.

5 Discussion

Machine invention of concepts within unrestricted first order Horn clause logic is at least as ambitious as most other problems within Artificial Intelligence. For this reason progress is likely to be slow. However, the method of inverting resolution described in [17] provides a logical basis for a very general form of inductive inference. This paper describes an information

theoretic approach to constraining the hypothesis space for inverse resolution, and an attempt is made to integrate the logical, computational and information-based aspects of hypothesis formation.

Many obstacles lie ahead in the further development of the inverse resolution approach to machine learning. These include

1. **Noise.** CIGOL works on the unacceptable assumption of completely noise-free data.
2. **Time.** CIGOL does not take into account the time complexity of executing the hypotheses which it forms.
3. **Unrestricted operators.** The derivation of the inverse resolution operators in [17] used a number of assumptions to simplify the derivation. These assumptions restrict the CIGOL operators unduly.

The work described in sections 3.3 and 3.4 of this paper may give us some lead on the problem of noise. This comes from the definition within algorithmic information theory of random (incompressible) strings. Indeed incompressibility is one of the most ubiquitous features that distinguishes noise from signal. Thus the incompressibility of sections of a set of examples would be a strong indication that the corresponding examples are noisy. It remains to be seen whether the relationship between noise and compressibility might be put to use within a system such as CIGOL.

References

- [1] Banerji, R. (1987). Learning in the limit in a growing language. In *IJCAI-87* Los Angeles, CA, Kaufmann, pp. 280-282.
- [2] Bennett, C. (1988). Logical depth and physical complexity. In *The Universal Turing Machine A Half Century Survey*, R. Herken, Ed., Kammerer and Unverzagt, Hamburg, pp. 227-257.
- [3] Buntine, W. (1988). Generalised subsumption and its applications to induction and redundancy. *Artificial Intelligence* 36, 2, 149-176.
- [4] Chaitin, G. (1987). *Information, Randomness and Incompleteness - Papers on Algorithmic Information Theory*. World Scientific Press, Singapore.

- [5] DeJong, G. (1981). Generalisations based on explanations. In *IJCAI-81*, Kaufmann, pp. 67-69.
- [6] Epstein, S. (1987). On the discovery of mathematical theorems. In *IJCAI-87* Los Angeles, CA, Kaufmann, pp. 194-197.
- [7] Gödel, K. (1962). *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*. Oliver and Boyd, London.
- [8] Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Prob. Inf. Trans.* 1, 1-7.
- [9] Langley, P., Bradshaw, G., and Simon, H. (1983). Rediscovering chemistry with the Bacon system. In *Machine Learning: An Artificial Intelligence Approach*, R. Michalski, J. Carbonnel, and T. Mitchell, Eds., Tioga, Palo Alto, CA, pp. 307-330.
- [10] Lenat, D. (1981). On automated scientific theory formation: a case study using the AM program. In *Machine Intelligence 9*, J. Hayes and D. Michie, Eds., Horwood, New York.
- [11] Michalski, R. (1983). A theory and methodology of inductive learning. In *Machine Learning: An Artificial Intelligence Approach*, R. Michalski, J. Carbonnel, and T. Mitchell, Eds., Tioga, Palo Alto, CA, pp. 83-134.
- [12] Michalski, R., and Larson, J. (1978). *Selection of most Representative Training Examples and Incremental Generation of VL1 Hypotheses: the underlying Methodology and the Description of Programs ESEL and AQ11*. UIUCDCS-R 78-867, Computer Science Department, Univ. of Illinois at Urbana-Champaign.
- [13] Michalski, R., and Stepp, R. (1983). Learning from observation: conceptual clustering. In *Machine Learning: An Artificial Intelligence Approach*. R. Michalski, J. Carbonnel, and T. Mitchell, Eds., Tioga, Palo Alto, CA, pp. 331-364.
- [14] Mitchell, T. (1982). Generalisation as search. *Artificial Intelligence* 18, 203-226.
- [15] Mitchell, T., Keller, R., and Kedar-Cabelli, S. (1986). Explanation-based generalization: a unifying view. *Machine Learning* 1, 1, 47-80.

- [16] Muggleton, S. (1987). Duce, an oracle based approach to constructive induction. In *IJCAI-87*, Kaufmann, pp. 287-292.
- [17] Muggleton, S., and Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. In *Machine Learning 5*, Kaufmann, pp. 339-352.
- [18] Niblett, T. (1988). A study of generalisation in logic programs. In *EWSL-88* London, Pitman.
- [19] Plotkin, G. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, 1971.
- [20] Popper, K. (1972). *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge and Kegan Paul, London.
- [21] Quinlan, J. (1979). Discovering rules from large collections of examples: a case study. In *Expert Systems in the Micro-electronic Age*, D. Michie, Ed., Edinburgh University Press, Edinburgh, pp. 168-201.
- [22] Rendell, L. (1985). Substantial constructive induction using layered information compression: tractable feature formation in search. In *IJCAI-85*, Kaufmann, pp. 650-658.
- [23] Robinson, J. (1965). A machine-oriented logic based on the resolution principle. *JACM* 12, 1, 23-41.
- [24] Sammut, C., and Banerji, R. (1986). Learning concepts by asking questions. In *Machine Learning: An Artificial Intelligence Approach*. Vol. 2, R. Michalski, J. Carbonnel, and T. Mitchell, Eds., Kaufmann, Los Altos, CA, pp. 167-192.
- [25] Shannon, C., and Weaver, W. (1963). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana.
- [26] Solomonoff, R. (1964). A formal theory of inductive inference. *J. Comput. Sys.* 7, 376-388.
- [27] Wirth, R. (1988). Learning by failure to prove. In *EWSL-88* London, Pitman.
- [28] Wrobel, S. (1988). Automatic representation adjustment in an observational discovery system. In *EWSL-88* London, Pitman.